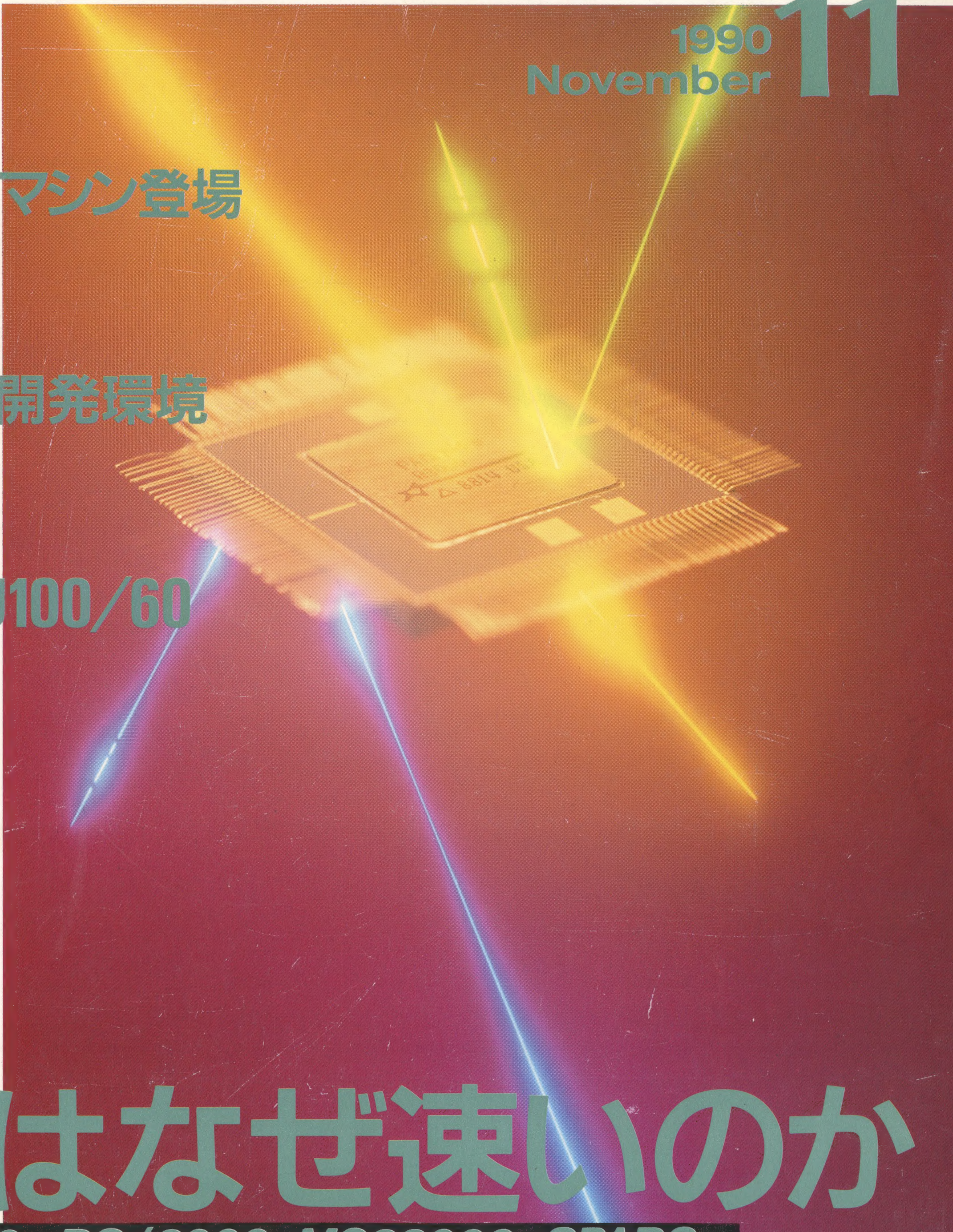
平成2年11月1日発行（毎月1回1日発行） 第1巻 第4号 通巻4号

コンピュータ技術総合誌 月刊スーパーアスキー

Super ASCII

FOR ADVANCED COMPUTER USERS

1990 November 11



Super EXPRESS

25MHzの68040を搭載した

5000ドルのNeXTマシン登場

INSIDE SOFTWARE

SDKによるアプリ開発の全貌

Windows 3.0の開発環境

TEST LAB.

高性能98はどこへ向かうのか

PC-H98model U100/60

開発環境が大幅に充実した

HyperCard 2.0

新連載

Emacsを過激にカスタマイズする

Emacs Lisp入門

特集

RISCはなぜ速いのか

R3000 RS/6000 MC88000 SPARC

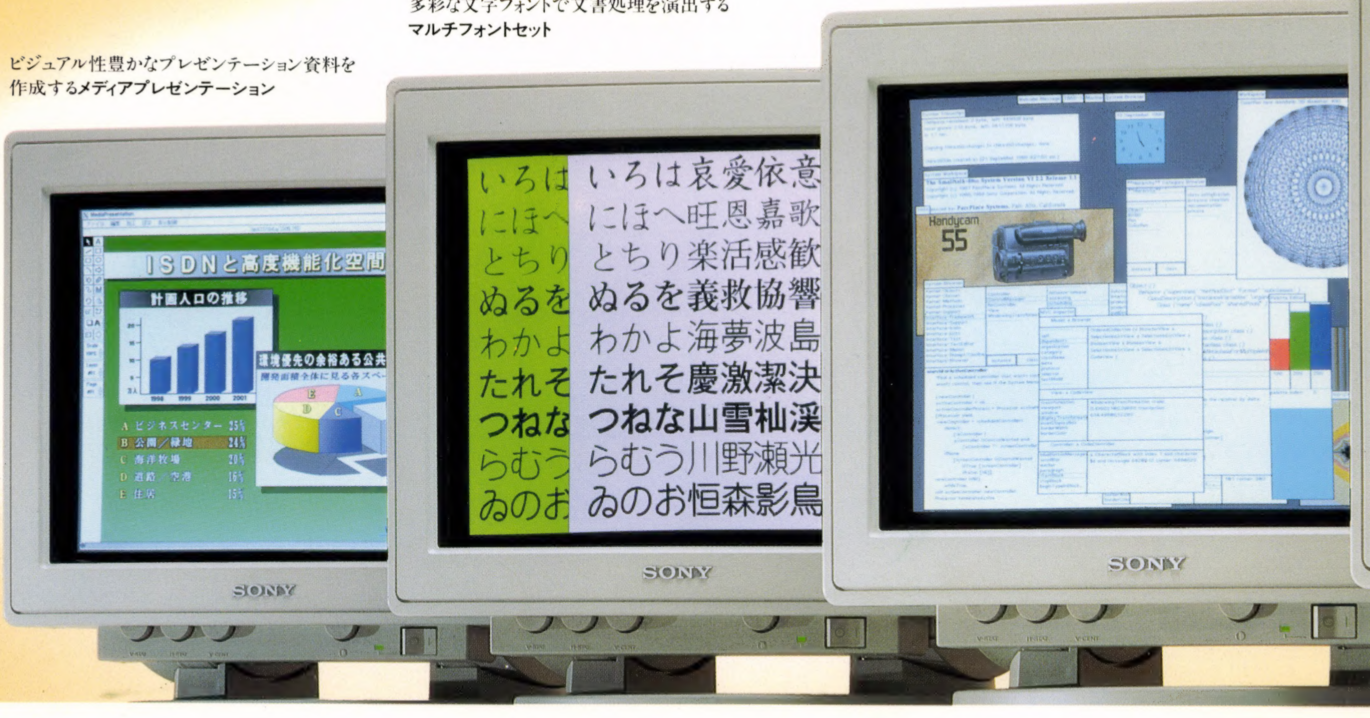
SONY

実現、高速25MIPS、大容量1.25Gバイト。NEWSは、新領域へ。

オブジェクト指向言語として定評ある
NEWS Smalltalk-80

多彩な文字フォントで文書処理を演出する
マルチフォントセット

ビジュアル性豊かなプレゼンテーション資料を
作成する**メディアプレゼンテーション**



広範なニーズに応える、充実のラインアップ。新たにOSも強化されました。

開放型分散処理の標準機として、ソフトウェア開発からDTP、CAD／CAM、そしてビジネスにいたる広範な分野で活躍する、ソニーのネットワークステーションNEWS。CPUにRISCプロセッサーを採用し、コンパクトなデスクトップサイズで高速処理を実現したRISC NEWSに新製品が加わり、ラインアップがさらに充実しました。25MIPSという高速処理を実現し、メインメモリー64Mバイト（最大128Mバイト）、大容量1.25Gバイトのハードディスクを搭載した最高峰モデル、NWS-3870。そして低価格ながら拡張性に富んだNWS-3700シリーズが登場。コンパクトなボディにRISCならではの高性能を凝縮した、NWS-3400シリーズと合わせ、3シリーズのラインアップです。また、RISC NEWSはすべてオーディオインターフェースを標準で搭載し、来るべきマルチメディア時代をも開拓。NEWSは常に先進のアーキテクチャーでワークステーションをリードしていきます。

NWS-3800シリーズ

NEWSの最高峰、NWS-3800シリーズ

- CPU：R3000
- コプロセッサー：R3010
- I/Oプロセッサー：MC68030
- MIPS値：25（NWS-3870）、20（NWS-3840、3860）
- メインメモリー：64MB／最大128MB（NWS-3870）、16MB／最大80MB（NWS-3840、3860）
- ハードディスク：1.25GB（NWS-3870）、640MB（NWS-3860）、286MB（NWS-3840）
- 拡張スロット：3
- オーディオインターフェース内蔵

ソニーは10月22日(月)より東京晴海で開催されるデータショウ'90に、NEWS、クォーターエル、プロデュースとデータディスクマンを出品します。

新たにデスクトップ環境、ネットワーク管理機能を搭載し、
ますます充実の**NEWS-OS**

テキスト/イメージ/音声のマルチメディアデータを
作成編集する**メディアクリエーター**

NWS-3870

NWS-3700シリーズ

**低価格ながら
拡張性に富んだ
NWS-3700シリーズ**

- ●CPU:R3000
- ●コプロセッサー:R3010
- ●MIPS値:17
- ●メインメモリー:16MB/最大128MB
- ●ハードディスク:640MB
- ●拡張スロット:3
- ●オーディオインターフェース内蔵

NWS-3720

NWS-3400シリーズ

**コンパクトなボディに高性能を凝縮した
NWS-3400シリーズ**

- ●CPU:R3000
- ●コプロセッサー:R3010
- ●MIPS値:17
- ●メインメモリー:8MB/最大16MB
- ●ハードディスク:415MB(NWS-3460)、ディスクレス(NWS-3410)
- ●拡張スロット:1
- ●オーディオインターフェース内蔵

NWS-3460

SAFS-25R

**高信頼性ファイルシステム
「SAFS」を装備した
ファイルサーバーシステム
「SAFS-25R」**

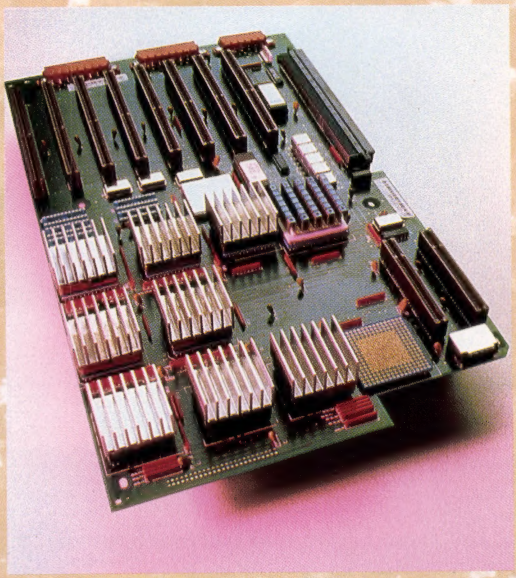
- ●CPU:R3000
- ●コプロセッサー:R3010
- ●I/Oプロセッサー:MC68030
- ●MIPS値:20
- ●メインメモリー:16MB/最大80MB
- ●ハードディスク:640MB×4
- ●拡張スロット:3
- ●オーディオインターフェース内蔵

お問い合わせ先 ソニー株式会社 NEWSインフォメーションセンター
〒141 東京都品川区東五反田1-22-1 五反田ANビル TEL(03)448-3553
※Smalltalk-80はゼロックス社の登録商標です。

開放型分散処理ワークステーション
NEWS

資料請求 No.001

C O N T



Super ASCII 11月号

November 1990 No.4

17 **Super EXPRESS**

●5000ドルを切ったNeXTのニューマシン——18

41 **特集**

RISCはなぜ速いのか

R3000 RS/6000 MC88000 SPARC ………太田昌孝

RISC CPUの基本的原理を説明し，その知識を前提に各社のRISCの特徴と性能を比較する．なぜ，あのようなアーキテクチャが使われるか，コンピュータそのものの成り立ちから説明する．

RISCとは何か／RISCの実現／CISC，RISC CPU基本性能一覧

65 **INSIDE SOFTWARE**

Windows 3.0の開発環境

● Windows 3.0 Software Development Kit……梅原　系/遠藤孝信

81 **TEST LAB.**

PC-H98model U 100/60——82
●高性能98はどこへ向かうのか

NWS-3460——88
●小型，低価格のRISC NEWS

HyperCard 2.0——92
●開発環境を大幅に充実

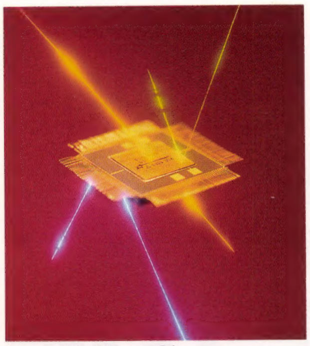


Photo:Yoshikazu Okamura

E　N　T　S

98 Hardware=Software+Hardware

400MIPSをめざす 並列処理コンピュータ Multimax

104 特別企画

第2回 PostScript Level 2の全貌……宮部義幸

連載

●システム探検隊 高エネルギー物理学研究所の巻……船田 巧――109
●Astro Guide '90 火星探査の新時代――112
●プロフェッショナルワークベンチとしてのUNIX入門 第4回 ニュースシステム……村井 純――113
●OS/2アーキテクチャ入門……宮澤篤之――119

プログラミングフォーラム

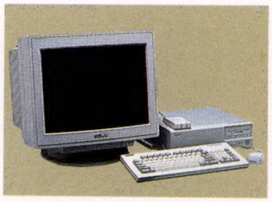
●386 de MS-DOS/プロテクトモードの管理……明石真平――126
●Emacs Lisp入門……半田剣一／小方一郎――132
●ウィンドウシステムのプログラミングパラダイム……中川幸男――138

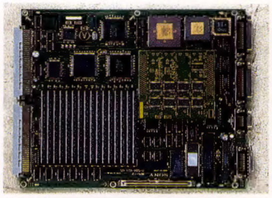
145 EP news

●基礎講座 文書交換のための共通文書フォーマットの動向 その3 MDI 日本DEC 企画本部システムインテグレーテッドグループ――146
●キーワード フォント表現フォーマット IKフォーマット INSTACフォーマット……大伴武都美／村山 登――155

161 MIPS

●パソコン1024夜……福崎俊博――162
●プログラマのための数学的世界観入門……斎藤 明――164
●ウサギと月と，そして太陽と……砂原秀樹――166
●Fat Bits……山崎俊一――168
●Book Shelf「電子出版」――170
●Readers Board――172
●最新ソフトウェアバージョンリスト――174


NWS-3460


PC-H98のCPUボード

NeXTのニューマシン

第2のNeXT。

NeXTstationがやってくる。
パーソナル・コンピューティングの時代が、いま新次元を迎える。
NeXTは、インターパーソナルコンピューティングへ。

人と人を結び、最新の情報をグループで共有し、より創造的な環境をつくるインターパーソナルコンピューティング。NeXTから始まった、1990年代のコンピュータ環境です。最先端のテクノロジーを凝縮したNeXTstationは、NeXTコンピュータが理想として掲げるコンセプトをすべて継承しつつ、パソコン並みの身近な価格を実現しています。薄型・コンパクト設計のNeXTstation本体には、モトローラの最新鋭68040(25MHz)CPUを搭載。驚異的な処理スピード、15MIPSを達成しました。また、2.88MBの3.5インチフロッピーディスクドライブを標準装備。MS-DOSフォーマットされた1.44MBまたは720KBフロッピーディスクの読み書きも可能で、既存のコンピュータ資源を無駄にしません。ハードディスクは105MB、RAMは8MBを標準実装。それぞれ340MB、32MBまで拡張できます。MegaPixel Displayは、作業中のページ全体をフルスクリーンで表示。"90年代のコンピュータにサウンドは不可欠"というコンセプトに基づいて、ディスプレイにマイクを内蔵しました。さらに、ハイパワー電子メール・NeXTmailを最初からバンドルし、Thin EthernetおよびTwisted-pair Ethernetを採用。あらゆるコンピュータにアクセスでき、多彩な情報を分散・統合する高度なマルチメディアネットワーク環境を構築できます。パワフルなUNIXのマルチタスク環境のもとで、スタンドアロンマシンとして、ネットワーキングコンピュータとして、十分なパフォーマンスを提供。一度使えば、もうNeXTを手放せなくなります。

資料請求 No.005

キヤノン株式会社 [お問い合わせは]キヤノン株式会社 ネクスト販売企画部 TEL.(03)344-8575・(044)549-5296/システム市場開発センター TEL.(03)757-6548/キヤノンソフトウエア株式会社 ネクスト販売事業部 TEL.(03)455-9920

sun
microsystems

みんな本当は コンピュータを 使いたいんだ。

〈データショウ'90出展のお知らせ〉

日時/平成2年10月22日(月)〜25日(木)9:30〜17:30

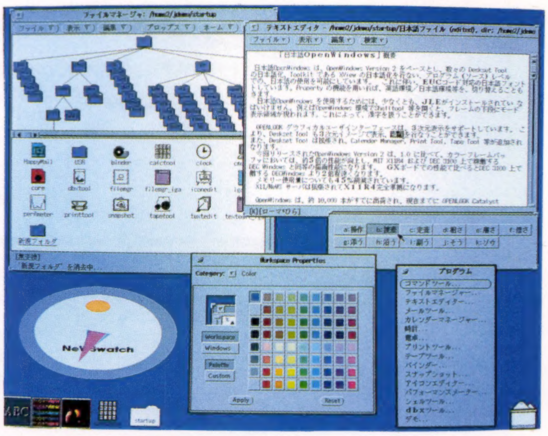
会場/東京国際見本市会場(晴海)・西館

お気軽にご来場ください。

（みんなが使える、創れる、自由なコンピュータ環境を。日本語OpenWindows誕生。）

UNIXの優れた能力を、ワークステーションやネットワークの高機能を、誰もが思いのままに使いこなせる。日本語で話し、考え、創る私たちと、コンピュータの創造力をスムーズに結ぶインタフェース、OPEN LOOK対応の日本語OpenWindows誕生。いま、人とコンピュータの間に、対話の窓が大きく開きます。

より多くの人々が、身近にあるさまざまなコンピュータを通して、もっと自由にコミュニケーションを図ったり、創造力を発揮できれば…。このような「オープン・コンピューティング」のコンセプトのもと、Sunでは、人とコンピュータが対話するためのウインドウ環境においても、より自由度の高い「日本語OpenWindows」をお届けします。「日本語OpenWindows」は、UNIX※システムなどのグラフィカル・ユーザ・インタフェース(GUI)として業界標準となりつつある「OPEN LOOK」※をベースに、アプリケーション開発ツール「XView」やウインドウ・システム「X11/NeWS」など、先進性と柔軟性を兼ね備えた3つの主要コンポーネントにより構成されています。さらに、ウインドウベースの使いやすいユーティリティを集めた「DeskSet」や、対話形式の開発者向けツール「日本語Devguide」も用意。アプリケーションやコンピュータの種類を問わない統一されたGUIにより、エンドユーザの方には、初心者でも、多彩な日本語アプリケーションを画面上のシンボル操作などで簡単に使いこなせる、容易な操作性を。また、ソフトウェア開発者の方には、ユーザ・インタフェースの設計・開発にかける時間の大幅な短縮による、生産性のより高い開発環境を、それぞれ提供することが可能です。現在、ハードウェア・メーカーやソフトウェア開発会社など多数の企業が、「日本語OpenWindows」の支持や採用を表明。Sunでは、すでに定評をいただいているSPARCstationの低価格・コンパクト化とともに、この「日本語OpenWindows」の提供を通して、UNIXの優れた能力、ワークステーションやネットワークの高機能を誰もが思いのままに使える、真に開かれたコンピュータ環境を目指します。



※UNIXは、米国AT&T社が開発しライセンスしています。※OPEN LOOKは、米国AT&T社の登録商標です。

日本サン・マイクロシステムズ株式会社

本社	〒102	東京都千代田区二番町11-19興和二番町ビル	☎(03)221-7021(代)
大阪支社	〒550	大阪市西区京町堀1-4-16センチュリービル9F	☎(06)445-5231(代)
名古屋営業所	〒460	名古屋市中区錦3-6-29興和ビルサウスハウス7F	☎(052)963-8560(代)
つくば営業所	〒305	茨城県つくば市春日3-12-10高野ビル2F	☎(0298)51-2210(代)

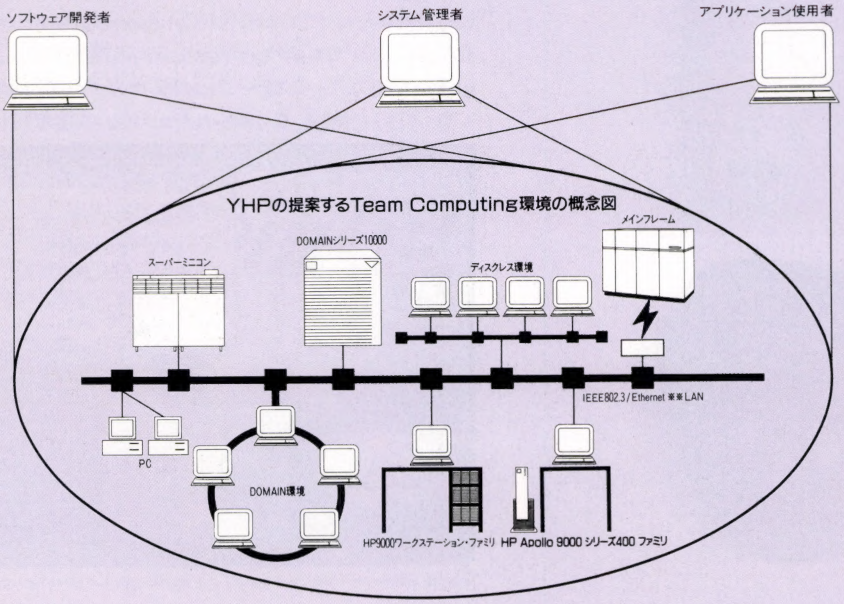
■ご希望の方に広告の商品のカタログを差し上げます。右の請求券を貼り、お名前、ご住所、お勤め先、部署、お電話番号を明記の上、ハガキ、またはFAXで下記までお送りください。〒102 東京都千代田区二番町11-19興和二番町ビル FAX(03)221-1974日本サン・マイクロシステムズ株式会社マーケティング本部まで。

日本語OW
請求券
SA

資料請求 No.007

アプリケーション変わる。

世界の200社以上がDCE(分散コンピューティング)の中核としてHPのNCSを採用。
(Network Computing System)



YHPの〈チーム・コンピューティング〉が変える。

ネットワーク上のハードやソフト、さらにはその操作法までも統合。ネットワーク全体をひとつのスーパーコンピュータとして機能させる〈チーム・コンピューティング〉の発想に、いま世界が注目しています。それは、誰もがそれぞれの目的にあわせて、スーパーなコンピュータ・パワーを優しく駆使できる、真に開かれた異機種分散ネットワーク環境。アプリケーションを開発するあなたにも、かつてない進歩をもたらします。

すべてのアプリケーション開発者のために。

アプリケーションを、もっと自由にのびのびと開発したい。〈チーム・コンピューティング〉は、アプリケーション開発者の願いや期待にも、しっかりと応えます。自由で制約のないCPU環境。すでに開発したアプリケーションをより強力に機能させる環境。さまざまな作業を優しく効率よく支援する充実のツールなど。そうした環境がもたらす進歩をご紹介しましょう。

世界が、未来標準に選んだ。

YHPのTEAM COM

開発が

◎オブジェクト・オリエンテッドな操作環境の実現：何をしたいかをアイコンなどで指示すると、それに伴う複雑な手続きが次々とディスプレイに表示され、アプリケーション開発の効率を高めることができます。

◎理想的なCPU利用環境を実現：ネットワーク上のCPUの中から、目的に最もふさわしいCPUを選んで利用できます。このためパソコンやワークステーションのCPU能力を意識することなく、強力でスケールの大きなアプリケーションを開発できます。

◎異機種間での操作統一を実現：OSF/[*]Motifと共通のユーザインタフェースによって、パソコンからスーパーコンピュータまでの操作法を統一。コンピュータにあまり馴じみのない方も、優しく接することのできるコンピューティング環境を実現します。

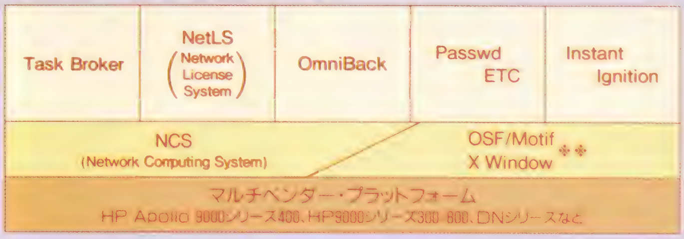
◎既存アプリケーションの活用を実現：すでに導入しているアプリケーションを、分散コンピューティングのもとで利用できる環境を整備しています。

◎分散アプリケーション開発ツールが豊富：NIDLコンパイラや分散アプリケーション・デバッギング・ツール、NCS分散管理ツールなど、アプリケーションの開発を強力にサポートし、生産性を向上させるツールを豊富に揃えています。

〈チーム・コンピューティング〉は未来進行形。

〈チーム・コンピューティング〉のめざすもの。それは、人が主役のコンピューティング環境。その世界標準化をリードするYHPは、NCSやOSF/Motifに続く進歩を、これからも積極的に発信していきます。日々進歩し続ける〈チーム・コンピューティング〉のあすに、ご期待ください。

■Team Computingの製品構成



※OSFは、先進的なOSや、オープンで移植性に優れたアプリケーションの開発、仕様の統一などのために設立された財団です。HP Apolloは設立メンバーとして、積極的な活動をしています。

※※OSF/MotifはOSFの、X windowはMITの、EthernetはXerox社のそれぞれ登録商標です。

■HP Apollo 9000シリーズ400

HPとApolloがそれぞれの最先端テクノロジーを融合させ、統一後初めて具現化した最新鋭モデルです。◎90万円台の12MIPSのモデル400dlから超高速26MIPSのモデル433sまで18機種をラインアップ。DTPやソフト開発、CAE/CAD、CG、アニメーション開発など3200種以上のアプリケーションをサポート。◎MC68030およびMC68040プロセッサを採用。2つのOS、Domain/OSとHP-UXを選択できるので、アプリケーションなど既存の資産も保護できます。◎異機種分散コンピューティング環境を具現化した〈チーム・コンピューティング〉製品群をサポート。スタンドアロン感覚でネットワーク上のCPUパワーを駆使できるなど、随所にかつてない先進性を持たせています。

HP Apollo 9000シリーズ400　モデル400s/433s

HP Apollo 9000シリーズ400　モデル400t/425t

データショウ'90出展
2年10月22日(月)～25日(木)
東京国際見本市会場（晴海）東館E-13

PUTING

資料請求 No.009

お問い合せはフリーダイヤルで
カストマ・コミュニケーションセンタ
0120-081444
YHP 横河・ヒューレット・パッカード
本社 〒168 東京都杉並区高井戸東3-29-21

AD-ACORP 0030

ICM

80MB 大容量が、常識です。130MB

というごとで、ついこの前まで20MBだ、40MBだと言っていたハードディスクも、今や少なくとも80MB、欲を言えば130MBが常識なのだそうです。これも知っていてソンはないと思いますよ、多分。

ハードディスクユニット

SRC-80/81/131
STRIDE
ストライド

「大は小をかねる」、機能面から見た常識です。さらに、「3足1000円」の靴下、数がまとまると単価がさがる。これも常識。誰もが知っていて当然で、しかも知らないとソンすること、それが常識。

SR-80
- 平均シーク28mS高速型80MBハードディスクユニット
- 40MB×2モード、80MB×1モードで使用できます。
- 80MB×1モードは、MS-DOS Ver.3.3/3.3A/3.3B/3.3Cのノーマルモードで使用できます。
- ハイレゾリューションモード及びOS/2では、40MB×2モードのみ使用可能です。

SRC-80
- 平均シーク28mS高速型80MBハードディスクユニット
- SASI/SCSI両コマンドモードが選択できます。
- SASIコマンドモード時は、80MB×1及び40MB×2モードで使用できます。また、80MB×1モードは、MS-DOS Ver.3.3/3.3A/3.3B/3.3Cのノーマルモードで使用できます。さらに、ハイレゾリューションモード及びその他のOSでは、40MB×2モードのみ使用可能です。
- SCSIコマンドモード時は、80MB×1モードでのみ使用可能です。

SRC-81
- 平均シーク28mS高速型80MBハードディスクユニット
- SCSIバス接続型専用機。80MB×1モードでのみ使用可能です。

SRC-131
- 平均シーク19mS高速型130MBハードディスクユニット
- SCSIバス接続型専用機。130MB×1モードでのみ使用可能です。

SR-40
- 平均シーク29mS高速型40MBハードディスクユニット
- 40MB×1モード及び20MB×2モードで使用できます。

SRC-40
- 平均シーク28mS高速型40MBハードディスクユニット
- SASI/SCSI両コマンドモードが選択できます。
- SASIコマンドモード時は、40MB×1及び20MB×2モードで使用できます。
- SCSIコマンドモード時は、40MB×1モードでのみ使用可能です。

SRC-41
- 平均シーク28mS高速型40MBハードディスクユニット
- SCSIバス接続型専用機。40MB×1モードでのみ使用可能です。

SR-80	SRC-80	SRC-81	SRC-131	SR-40	SRC-40	SRC-41
¥168,000	¥158,000	¥158,000	¥198,000	¥118,000	¥108,000	¥108,000

■SRシリーズは、インターフェイスカード、ケーブル、ユーティリティ(5"-2HD)付きです。
■SRCシリーズは、ケーブル、ユーティリティ(5"-2HD)付きインターフェイスカードは別売です。
■SRC-40/80は、標準出荷時は、SASIモードでフォーマットされていますから、SCSIドライブとして使用する場合は、再フォーマットが必要です。

【IF-2720】PC-9800及びPC-286シリーズ用 SASIコマンド用ハードディスクインターフェイスカード……¥20,000
【IF-2750】PC-9800及びPC-286シリーズ用 SCSIコマンド用ハードディスクインターフェイスカード……¥30,000

＊表示価格には消費税は含まれておりません　＊MS-DOSは米国マイクロソフト社の登録商標です

ICM 株式会社アイシーエム

- 本社営業部/〒556 大阪市浪速区日本橋4丁目2番9号 TEL. 06(648)4702 FAX. 06(647)2018
- 東京営業所/〒101 東京都千代田区神田小川町2丁目2番サンブリチビル2F TEL. 03(291)7491 FAX. 03(291)7494
- 名古屋営業所/〒450 名古屋市中村区名駅南2丁目14番19号住友生命名古屋ビル10F TEL. 052(561)4927 FAX. 052(561)3644
- 北海道営業所/〒060 札幌市中央区北1条西2丁目1番地時計台ビル11F TEL. 011(210)4778 FAX. 011(210)4766
- 開発・企画・営業スタッフ募集中/お気軽にお問い合わせ下さい(総務部06-647-2008まで)

ワークステーションのPFUへ。

パソコン、オフコン、ミニコン PFU

なくてもできる。

ソフトウェア開発者の皆さん、グッドニュースをお届けします。PFUのEPOCファミリでMS®OS/2ソフトの開発が、OSや言語に熟知していなくてもできるようになりました。Cシリーズを使って高度な処理が簡単な操作で実現できます。

❶MS®OS/2の勉強をしなくてもできる。

EPOACE/2は、日本語MS®OS/2(V1.1)上で作動する事務処理用の簡易ソフト。OSを意識せず、会話形式で簡単にアプリケーション開発ができます。

❷C言語のプロでなくてもできる。

プログラム言語を熟知したプロでなくてもMS®OS/2対応の高度なアプリケーションを開発できます。また、MS®OS/2の共通規約をチェックする作業も不要になります。

❸コンバージョンをしなくてもできる。

従来のEPOACEで開発したアプリケーションは、そのまま日本語MS®OS/2上で使用できます。OSが、CASやMS-DOS™から変わっても、コンバージョンは一切不要です。

ますます高度化する情報活用を、強力にサポートするCシリーズのMS®OS/2。さらに、従来の資産を継承しながらMS®OS/2に対応した本格的な簡易ソフトEPOACE/2、EPOCALC/2、OASYS/2を提供。あらゆるビジネスシーンに柔軟に対応します。

MS®OS/2アプリケーションが、今日からつくれる。

事務処理汎用パッケージ

EPOACE/2

PFU Cシリーズ
C-285LT C-285RX
C-385RX

「EPOACE/2」開発コンテスト実施中！ お問い合わせ先：東京本社営業推進部販売促進課 ☎03-438-0311

※MSは米国マイクロソフト社の米国での登録商標です。
※EPOACE、EPOCALCは登録商標です。
※OASYSは富士通㈱の商標です。※MS-DOSは米国マイクロソフト社の登録商標です。

株式会社 PFU

東京本社/営業部 〒105 東京都港区西新橋3-25-33(NP御成門ビル)☎(03)438-0311
大橋分室 〒153 東京都目黒区大橋2-16-27(飛栄大橋ビル)☎(03)467-0121

札幌営業所/〒065 札幌市東区北11条東6-29-26 ☎(011)723-6761
神奈川営業所/〒242 神奈川県大和市深見西2-7-11 ☎(0462)63-1720
北陸営業所/〒920 金沢市広岡2-13-33(K.R.D.ビル)☎(0762)60-4751
名古屋営業所/〒460 名古屋市中区錦1-19-25(東京生命館)☎(052)202-0871
大阪営業所/〒530 大阪市北区堂島1-5-17(堂島グランドビル)☎(06)344-3648
広島営業所/〒730 広島市中区基町12-3(朝日生命広島ビル)☎(082)228-3551
福岡営業所/〒812 福岡市博多区博多駅前1-5-1(朝日生命福岡ビル)☎(092)472-7056
長崎営業所/〒852 長崎市平野町1-35(坂上ビル)☎(0958)43-7121
Uプラザ/〒105 東京都港区西新橋3-24-9(飯田ビル)☎(03)434-5446
EPOCプラザ/〒163 東京都新宿区西新宿2-4-1(新宿NSビル・OAセンター内)☎(03)342-0280

資料請求 No.011

NKKから3次元

エンタテーメントCG
〈CGアート/CGアニメーション〉

インダストリアルCG
〈建築パース/商品デザイン〉

パーソナルからスーパーグラフィック
各種アプリケーション・ソフトウェア

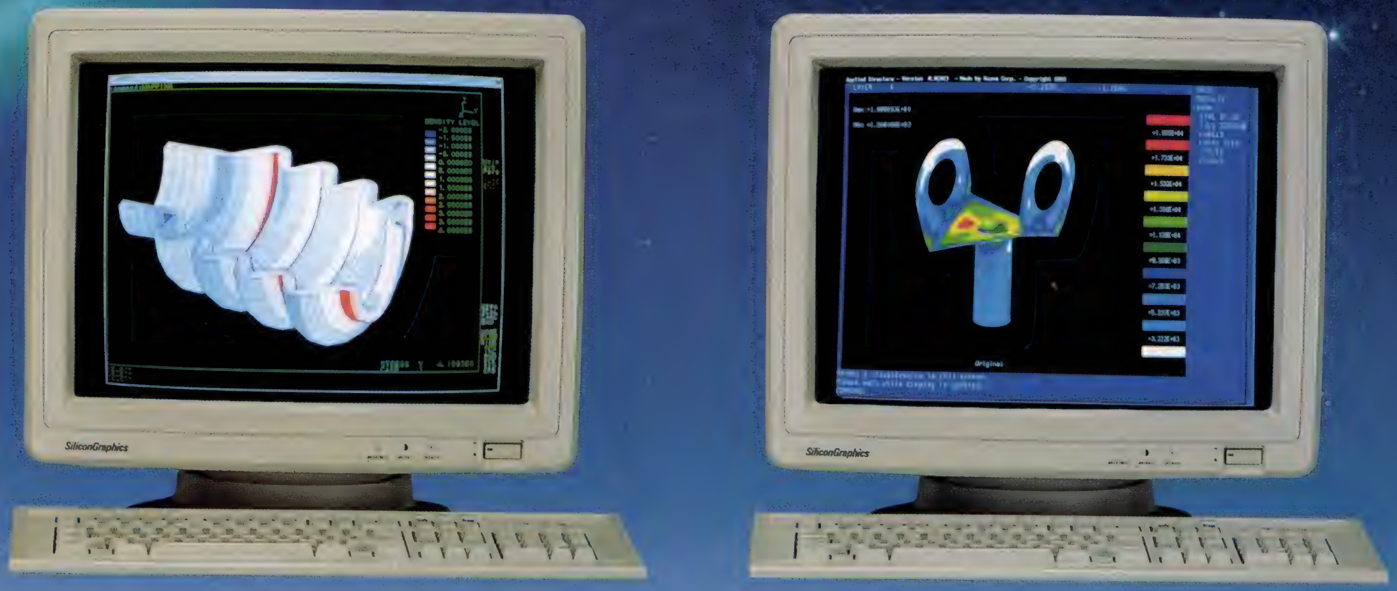
SiliconGraphics
IRIS-4D Family

●IRIS-4D/25TG

●IRIS-4D/340VGX

●IRIS-4D/380S(サーバー)

ワークステーション、発売中！



金型内樹脂流動解析
〈MOLDIA-F〉

P法有限要素法構造解析
〈ラズナ社APPLIED STRUCTURE〉

まで豊富なラインナップと
をサポート。

1,000,000ポリゴン/秒
1,000,000ベクター/秒の高スピードを実現。

POWERVISIONは、各種アプリケーション・ソフトウェアのパフォーマンスを最大限に引出します。

●Applied Structure（ラズナ社製）
P法有限要素法をベースとしてアダプティブな精度管理による解析・感度解析・最適設計を一体化した画期的なシステムがIRIS上で稼動、NKKより販売・サポートを開始します。

●MOLDIA-F
NKKで開発した射出成形金型内樹脂充填過程シミュレーションプログラム。成形不良の予測・検出が可能です。

●ADINA
有限要素法汎用構造解析システム。機能としては、線形/非線形の静的・動的構造解析、定常・非定常熱解析およびプリ・ポスト。

●コンピュータ グラフィックス、その他
NKKでは、他にもIRIS上で稼動する各種アプリケーション・ソフトウェアを販売、サポートしています。

●IRIS 4D25TG 16MIPS 1.6MFLOPS 200,000ベクター/秒 20,000ポリゴン/秒
●IRIS 4D340VGX 117MIPS 20.8MFLOPS 1,000,000ベクター/秒 1,000,000ポリゴン/秒
●IRIS 4D380S（サーバー） 234MIPS 32.7MFLOPS

4D/340VGX、4D/380SにはPOWER path パラレルRISCアーキテクチャとパラレル化されたUNIXのIRIXを採用し、並列処理を実現しています。

※UNIXはAT&T社の登録商標です。APPLTED STRUCTUREはラズナ社の登録商標です。
ADINAはADINA R&D社の登録商標です。IRIS-4Dはシリコングラフィックス社の登録商標です。

NICOGRAPH'90出展
会期 1990年11月6日(火)〜9日(金)
会場 サンシャインシティ・文化会館4F
コマ番号 総合ゾーン No.58

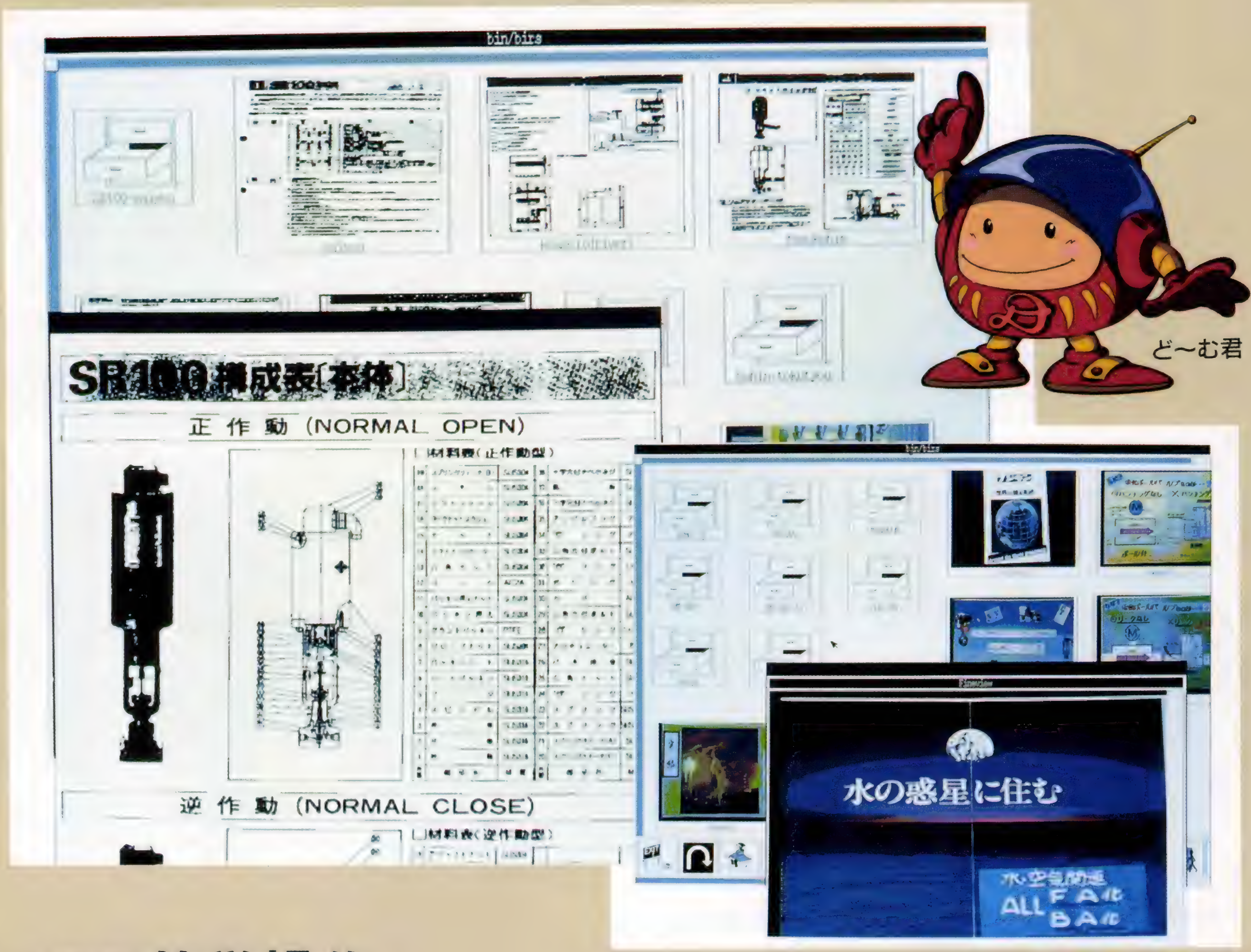
販売代理店

NKK
日本鋼管株式会社

エレクトロニクス本部 コンピュータ営業部
〒102 東京都千代田区九段北4丁目1番3号 飛栄九段北ビル
電話：(03)288-3830・3831（ダイヤルイン）
FAX：(03)288-3869

フジキンソフト株式会社

画像データベース



マウスで簡単操作

☞ アイコンやプルダウンメニューをクリックするだけで簡単に操作することが出来ます。

ツリー構造ですぐに検索

☞ 画面上に表示されている階層構造で、必要データをすぐにとり出すことが出来ます。

取り込んだ画面の編集・修正が可能

☞ 縮小・拡大・切り貼り・回転・合成・反転などの編集機能があります。

外部からの入力可能

☞ イメージリーダやビデオカメラから画像の入力がおこなえます。

ソニーNEWS 好評発売中

NWS-3460

■御用命は下記 フジキンソフト/フジキンまで

RISC NEWS
NEWS
ソニーパーソナルコンピューター Quarter L

フジキンソフトセンターセミナー日程表

※日程は都合により変更される場合があります。

テーマ　月	10月	11月	12月
NEWS(UNIX)入門	3日(水) 24日(水)	6日(火) 28日(水)	5日(水) 19日(水)
エディター入門(イメージパートナー)	2日(火) 25日(木)	27日(火)	4日(火) 20日(木)
NEWS(UNIX)C言語入門	16日(火) 17日(水)	20日(火) 21日(水)	11日(火) 12日(水)
MS-DOSユーザーのためのNEWS(UNIX)入門	4日(木)	7日(水)	6日(木)
X-Window入門	18日(木)	22日(木)	13日(木)
ネットワーク入門	9日(火)	8日(木)	18日(火)

▶その他のコース　前述以外に、各社のアプリケーションの応用講習会をメーカー、ソフトハウス並びにシステムハウスとタイアップして開催致します。

ASCIIとタイアップ

㈱アスキーInformix講習会

テーマ	10月	11月	12月
SQL入門	11日(木)、22日(月)	1日(木)、15日(木)	3日(月)
informix-SQL I	12日(金)、23日(火)	2日(金)、16日(金)	4日(火)

●講師/エキスパート・パワーオオサカ

◀アスキー主催の講習会です。詳しいお問い合わせは…
㈱アスキー システムソフトウェア事業部
営業推進部内 講習会係まで
TEL. 03-486-0821

コンピュータシステム事業部		
事業本部	TEL.06-376-4751	FAX.06-376-2342
EWSサポートグループ	TEL.06-337-5631	FAX.06-337-7540

フジキンは技術でお応えします！

ソニー株式会社 NEWS Quarter L 代理店

FUJIKIN SOFT
フジキンソフト株式会社
株式会社フジキン

東京本店	☎(03)273-0301	ソフト名古屋	☎(052)581-7861
芝中央営業所	☎(03)431-7841	名古屋営業所	☎(052)561-7681
八重洲営業所	☎(03)553-0381	名古屋中央営業所	☎(052)586-8011
京葉営業所	☎(0472)46-7201	関西支店	☎(06)532-5601
新宿中央営業所	☎(03)366-8181	阪神東営業所	☎(06)453-1871
横浜中央営業所	☎(045)662-4061	東大阪営業所	☎(06)787-2201
神奈川東営業所	☎(044)211-9351	京都中央営業所	☎(075)371-3241
日立営業所	☎(0294)24-2511	神姫営業所	☎(0794)25-3291
札幌中央営業所	☎(011)222-4681	山陽中央営業所	☎(0862)23-0441
仙台営業所	☎(022)265-3081	西部中央営業所	☎(092)291-0641
多摩中央営業所	☎(0423)25-6281	中国営業所	☎(082)263-2331
湘南中央出張所	☎(0467)87-3811	新居浜出張所	☎(0897)32-3811
筑波出張所	☎(0298)52-9021	大分出張所	☎(0975)38-2611
長岡中央営業所	☎(0258)34-1980	南九州出張所	☎(096)382-7591
中新潟出張所	☎(0258)36-7878	新大阪テクニカルセンター	☎(06)337-5631
静岡出張所	☎(0542)81-6407		

Fuji Software Inc.

私たちの富士ソフトウエアはじつはナレッジファクトリーです。

モダンタイムスの工場は嫌いだでもソフトウエアが産業になるためには新しい工場をもつべきだと思う。

【経験者募集】
(研修制度あり)

【募集要項】●資格／高卒以上22才以上。ビジネスソフト経験者も歓迎●**職域**／SE・プログラマ・インストラクタ●**給与**／経験に応じて当社規定により優遇 平成2年度大卒初任給・188,000円 ●**待遇**／昇給年1回、賞与年2回、交通費全額支給、社会保険有、有給休暇有、社宅・独身寮完備●**勤務地**／横浜・鎌倉・三田・芝浦・亀戸・川崎・厚木・立川・八王子・我孫子・名古屋・大阪・仙台の各事業所●**勤務時間**／フレックスタイム制・週休2日 ●**応募方法**／履歴書(写真貼付)・職務経歴書(詳細記入)、随時受付●**面接日**／書類選考の上別途ご連絡いたします。●**送付先**／〒136 東京都江東区亀戸2—25—14立花アネックスビル2F 富士ソフトウエア株式会社 業務部宛 ※応募の秘密は厳守いたします。

会社概要:●**設立**／昭和45年5月●**資本金**／15億1,910万円(払込)●**社員数**／1,300名●**資本系列**／なし(独立系)●**売上高**／80億円(平成元年度)●**事業内容**／●UNIX及びMS OS／2を中心とした各種制御ソフトの開発●各種パッケージソフトの開発・販売(毛筆わーぷろ等)●OS／2関連の各書籍出版及び販売●MS OS／2セミナ「マイクロソフト・ユニバーシティ(MU)」開催(米国マイクロソフト社公認)●ハードウエア開発

株式店頭登録No.9749

FSI

富士ソフトウエア株式会社

お気軽にお問い合わせください。 フリーダイヤル 0120-132213 (担当 吉元)

資料請求 No.016

カスタムコミュニケータ

ORANG

人とシステム できること、たくさん。HZS

"ORANG"は…

- CAD図面(キャド情報)も 手書き図面(イメージ情報)も一括管理
- CAD図面のオンライン登録を実現 情報はリアルタイムで出力可能
- 高速専用装置(ファセット)により 登録・検索がワンタッチ
- 必要な情報を一挙に絞り込み 1枚/秒のスピード検索が可能
- 自在なシステム構成が可能

"図面情報戦略ステーション"です

エンジニアリング業界に最適!
技術情報を一括管理する
進化した光ファイリング統合システムです。

標準システム

HZS
システムステーション

Hitachi Zosen Information Systems Co.,Ltd.

東京本社/〒108 東京都港区高輪1-4-10 ☎(03) 447-5300
大阪本社/〒550 大阪市西区江戸堀1-22-17 ☎(06) 448-5081
札幌営業所/〒060 札幌市北区北七条西1-1-2 ☎(011)707-8911
新潟営業所/〒940 長岡市東坂之上町2-1-1 ☎(0258)32-8495
北陸営業所/〒930 富山市総曲輪1-7-15 ☎(0764)33-6066
名古屋営業所/〒460 名古屋市中区丸の内3-18-28 ☎(052)961-0241
岡山営業所/〒700 岡山市柳町1-1-27 ☎(0862)32-2041
広島営業所/〒732 広島市東区光町1-9-28 ☎(082)262-9050
九州営業所/〒810 福岡市中央区天神2-14-8 ☎(092)761-7643
CADセンター/〒231 横浜市中区蓬莱町2-4-1 ☎(045)243-6471

資料請求 No.00 A

ASCII MAGAZINE

マイクロ
コンピュータ
総合誌

ASCII

[月刊アスキー] 11月号　10月18日発売　定価580円(税込み)

特集 聴くだけの音楽じゃつまらない
これから始める **コンピュータミュージック**

コンピュータミュージックは作曲家とミュージシャンだけのものではない!

「楽しい聴き方」から始めるコンピュータミュージック遊びを追究。第2部では3人の音楽家の現場を直撃した。

●でも、まずは聴くことから始めたい●音楽データで遊ぶ●挫折しないミュージくんライフ●コンピュータミュージック向け電子楽器選び●MINDLIGHT7の挑戦

特別企画 オーサリングシステムの世界
AVプログラミング講座 スプライトプログラミング実践編

ノート型パソコンの最新情報を徹底フォローする **ノート道(のーとみち)** [第2回]

●98NOTEのRAM増設法●ユーザーの声:山田祥平氏●MaxyNote286試用レポート●フリーウェア

初心者からパワーユーザーまで、幅広いニーズに応えるアスキーの雑誌

ASCII
MAGAZINE
INFORMATION

UNIX MAGAZINE ユニックス・マガジン

UNIX REVIEW誌提携　11月号
10月18日発売定価810円(税込み)

●Nemacsから使うメールシステム
●UNIX Communication Notes……telnet
●UNIX流プログラミング
●HP SoftBench環境
●日本語OpenWindowsバージョン2
●ワークステーション・ガイド…MUSBUS

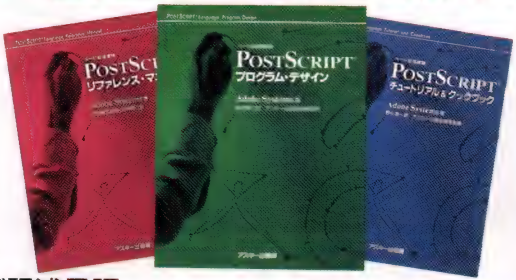
〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル(株)アスキー書籍/雑誌営業部 TEL(03)486-1977 **株式会社アスキー**

●ブックカタログ送呈:住所・氏名・年齢・職業・電話番号・使用機種を明記の上、宣伝部SAS係までハガキでお申し込みください。

資料請求 No.00 B

ASCII BOOK

アドビ社純正マニュアル3部作完成!



ページ記述言語

PostScript プログラム・デザイン

最新刊

アドビ・システムズ著 松村邦仁訳 アスキー出版技術部監訳 定価3,090円

PostScriptにおける、高速なプログラミングとは/効率のよいスタックや辞書の使い方とは/他のプログラムと協調するためのモジュール化の技法/真のPostScript対応アプリケーションと呼べるプログラムとは…/などの疑問に、豊富なソースコードによって答える。

ページ記述言語

PostScript リファレンス・マニュアル

アドビ・システムズ著 松村邦仁訳 石田晴久監修 定価3,090円

ページ記述言語

PostScript チュートリアル&クックブック

アドビ・システムズ著 野中浩一訳 アスキー出版技術部監修 定価3,090円

プログラミングOS/2 III・IV

PMプログラミング編〈上、下〉

チャールズ・ペゾルド著 飯倉弘一監訳 上巻定価5,800円、下巻定価3,500円

OS/2標準グラフィック環境

OS/2標準グラフィック環境であるプレゼンテーション・マネージャ(PM)のプログラミングを、基礎から、段階的に、豊富なサンプルプログラムを紹介しながら解説する。下巻では特に、ビットマップ/アイコン/ポインタ/テキストリソース/メニューとキーボードアクセラレータ/ダイアログボックスについて詳細に解説。

DISK ALBUM 41

プログラミングOS/2 III IV PMプログラミング編〈上、下〉

- ■MS OS/2 Ver. 1. 1以上, 日本語MS OS/2 Ver. 1. 1以上
- ■メディア:5-2DDおよび3.5-2DDの両方を同梱
- ■価格5,800円(税抜き)

入門UNIX

羽山 博著 定価1,800円

UNIXラーニングシステム 全3巻 刊行開始!

UNIXラーニングシステム第1弾の〝入門書〟UNIXに初めて触れる人、UNIXに習熟したい人を対象に、電子メールのやりとりや、画面エディタの使い方をまじえつつ、UNIXシステムの仕組みや操作、コマンドオペレーションをやさしく解説。System V版/BSD版ともに対応。

PC-H98 model 70 テクニカルデータブック

アスキー第二書籍編集部編 定価7,000円

PC-H98 model 70のテクニカル情報を詳解

NESAバスやハイレゾリューショングラフィックス、高速DMA転送やフルセントロプリンタ制御をはじめとして、ROM BIOSやサンプログラムなど、PC-H 98 model 70のテクニカル情報を網羅。ハードウェア・ソフトウェア開発ユーザー必読の内部情報を公開。

C言語を256倍使うための本

福崎俊博 梅原 系 山田伸一郎共著 定価1,100円

これでおなたも〝Cウィザード〟だ

Microsoft Cコンパイラをベースに、C言語プログラマの誰もがおちいりやすい落し穴、誰もが知りたいテクニックを一挙公開。アスキー「MS-DOS SOFTWARE TOOLS」を開発したメンバーが培ったノウハウをこの1冊に凝縮した。

VZを256倍使うための本

志村 拓 可部正三 鷲北 賢共著 定価1,100円

誰も書かなかったVZエディタ徹底活用術

使いこなしのノウハウをぎっしり凝縮した全VZユーザー必読の書。マウスやモデムを使いきる大規模マクロから、即実用で役に立つ小粋なマクロ、そしてゲームまで14本を大公開。あなたのレベルを確認できる「VZエディタ256倍度チェック」付き!

DISK ALBUM 40

VZを256倍使うための本

- ■メディア:5-2HC/3.5-2DD ■MS-DOS版
- ■対象機種:PC-9801/J-3100/AX(IBM PCは除く)
- ■価格3,500円(税抜き)

- ●PostScriptは、アドビ・システムズ社の商標です。
- ●UNIXオペレーティングシステムは、AT&Tのベル研究所が開発し、AT&Tがライセンスしています。
- ●MS-DOSは、米国マイクロソフト社の商標です。
- ●(税抜き)表示のないものは、すべて消費税込みです。

〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル㈱アスキー書籍/雑誌営業部 TEL(03)486-1977 **株式会社アスキー**

●ブックカタログ送呈:住所・氏名・年齢・職業・電話番号・使用機種を明記の上、宣伝部SAS係までハガキでお申し込みください。

資料請求 No.00C

ASCII
SEMICONDUCTOR

どんどん成長してるんだね。
〈ASCII〉×〈CHIPS & TECHNOLOGIES〉

CHIPS

IBM-PC/ATやPC/XTコンパチブルといった互換機用チップセットの開発で
リーディングカンパニーとして名をはせる「チップス＆テクノロジーズ」。
アスキーセミコンダクターはその日本代理店として優れた製品の数々を皆様に紹介し、
パーソナルコンピュータの世界に新たな可能性を創造してきました。
そして、21世紀に向けて、私たちはこの世界をもっともっと活性化させてまいります。
※IBM-PC/ATとIBM-PC/XTは、International Business Machines社の登録商標です。

株式会社アスキーセミコンダクターは米国CHIPS＆TECHNOLOGIES, INC.(日本代理店)のほか国内外の半導体を高度な技術サポートとともに、供給しています。

株式会社アスキーセミコンダクター
〒107-24 東京都港区南青山5-11-5住友南青山ビル
TEL.03-486-9188 FAX.03-486-8805

資料請求 No.00D

郵便はがき

料金受取人払

赤坂局承認

2257

差出有効期間
平成4年6月
30日まで
(切手不要)

107-24

東京都港区
南青山6-11-1
スリーエフ南青山ビル

株式会社アスキー
広告部
「資料請求」係行

あなたの欲しい資料はすべて入手可能です.

- 資料請求カードは,掲載広告に関する資料集めです.どうぞお気軽にご利用ください.
- なお,ご希望の資料がお手もとに届くまで,1ケ月ほどかかります.あらかじめご了承ください.
- カードの記入方法は,184ページをご参照ください.

郵便はがき

料金受取人払

赤坂局承認

2264

差出有効期間
平成3年12月
31日まで
(切手不要)

107-24

(受取人)
株式会社アスキー
スーパーアスキー
編集部行

NOV

月刊スーパーアスキー編集部では,読者の皆様の声をお聞きして,内容をさらに充実したものにしたいと考えています.表面の空白部分には,お便りやイラスト,スーパーアスキーへのご意見・ご感想などを自由にお書きください.

1990年11月号 ASCII アンケートハガキ

●HBなどの濃い鉛筆でご記入ください。

1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
2	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
3	1 2 3 4 5 6 7 8 9 10 11 12 13 14
4	1 2 3
5	1 2 3 4
6	1 2 3
7	1 2 3 4 5 6 7 8 9 10 11 12 13
8	1 2 3 4 5 6 7 8 9 10 11 12 13 14
9	1 2 3 4 5 6 7 8 9 10
10	1 2
11	1 2 3 4 5 6 7 8 9 10 11 12 13

フリガナ

お名前

ご自宅の住所　〒　都道府県　市区郡　☎（　）

使用機種　自宅 メーカー 機種　勤務先 メーカー 機種

読者プレゼント希望No.（商品名）

アンケート記入方法は175ページをご覧ください。

毎月10日までに到着したハガキの中より，抽選で50名の方に図書券500円分をさしあげます（ただし抽選は「今月のプレゼント」を優先させます）。

資料請求カード ASCII 11月号（AK）1991年1月15日まで有効

ご面倒でも，フリガナ・送付先ご住所等，全項目にご記入ください。

※お問合せ先：TEL.03-797-3464広告部まで

フリガナ

姓　名

生年月日19　年　月　日

性別　男・女　年齢　歳

送付先ご住所（1.自宅　2.勤務先）　〒　TEL（　）　都道府県　区市郡

↑○で囲んでください。

勤務先名／学校名

所属・役職／学部

職種　1. コンピュータ関連技術者　2. 1以外の技術者　3. 営業職　4. 事務職　5. 管理職　6. 公務員　7. 学生　8. その他

使用機種　自宅 メーカー 機種名　勤務先 メーカー 機種名

No.	No.
No.	No.
No.	No.
No.	No.
No.	No.

カードの記入方法は，184ページをご参照ください。

Super EXPRESS

NOVEMBER 1990

20	オムロンが68040採用のLUNAを発売 ソニーがRISC NEWSに新機種を追加
21	日本サンが日本語OpenWindowsを開発 富士通がメインフレーム用UNIXを発売
22	ソニーが386SX搭載のノート型パソコンを発表
23	松下が業界最薄型のノート型パソコンを発売 沖電気がAX仕様のノート型パソコンを発売
24	モトローラが制御用32bitプロセッサを発表 モトローラが高速SRAMの新製品を発表 富士通がアウトラインフォント処理用LSIを発売
25	IBMが汎用コンピュータの新世代システムを発表
26	インテルとアプリックスがPC-9800用DVI開発プラットフォームなどを発表 Linotypeが日本語PostScriptRIPなど3製品を発表
27	マイクロソフトがMS C.6.0の日本語版を発表 リード・レックスがWindows対応のDTPソフトを発売 富士ゼロックス情報システムがPC-9800用Smalltalk-80を発売
28	CSKがUNIX用のリアルタイム・スプレッドシートを発売 オムロンが日本語Motifを発売 アステックがSaber-Cの日本総販売代理店に
29	エルゴソフトとYHPがEGWordをHP9000シリーズに移植 アドビがIllustratorをバージョンアップ
30	NewsFlash
38	Event Calendar

18 NeXTが68040搭載のワークステーションなどを発表

NeXTが68040搭載のワークステーションなどを発表

米NeXT Computer社は，CPUにクロック周波数25MHzの68040を使ったワークステーション「NeXT Computer System」新製品3機種を投入するとともに，クロック周波数33MHzのi860をグラフィックアクセラレータとして使用した1670万色表示可能なディスプレイボード「NeXTdimension」を開発，発表した．

NeXT Computer System 3機種は，15MIPS，2MFLOPSの性能を実現している．内訳は，①従来のNeXTマシンと同形の「NeXTcube」，②ピザボックス型の普及マシン「NeXTstation」，③NeXTstationのカラー版「NeXTstation Color」――の3機種．

■拡張性重視の「NeXTcube」

NeXTcubeは，幅・奥行き・高さとも305mm（1フィート）の従来機と同じ立方体型．メモリ・記憶装置・バスなどで，拡張性を重視した設計となっている．

メインメモリは標準で8Mbytesを搭載し，最大64Mbytesまで拡張が可能．外部記憶装置は，標準構成では2.88MbytesのDOSコンパチブルなフロッピーディスクドライブと105Mbytesのハードディスクドライブをそれぞれ装備しており，340Mbytes／660Mbytes／1.4Gbytesの各ハードディスクドライブ，256Mbytes光磁気ディスクドライブ，CD-ROMドライブを搭載することも可能となっている．拡張バスとしてNeXTバスを3スロット装備し，NeXTdimensionボードなどが使用できる．

DSPとして米Motorola社の56001を装備し，専用SRAMを標準で24Kbytes搭載している．専用SRAMは最大576Kbytesまで拡張できる．

ディスプレイは，解像度1120×832ドットの4階調表示可能な17インチモノクロディスプレイを採用した．

そのほかインターフェイスとして，ThinEthernet，Twisted pairEthernet，RS-423×2，SCSI-2，プリンタポート，DSPポートなどを装備している．

写真1　「NeXTcube」プラス「NeXTdimension」のシステムで，フルカラーグラフィックスの取り扱いが可能となった．

■5000ドルを切った普及機「NeXTstation」

NeXTstationは，標準構成で4995ドルという低価格の普及マシン．筐体は，397.8（W）×364.5（D）×64（H）mmのピザボックス型を採用した．

メインメモリは標準で8Mbytesを搭載し，最大32Mbytesまで拡張が可能．外部記憶装置は，標準構成で2.88Mbytesフロッピーディスクドライブと105Mbytesハードディスクドライブをそれぞれ装備し，340Mbytesハードディスクドライブを内蔵することも可能となっている．

DSPおよびディスプレイ，インターフェイスについては，NeXTcubeと同スペック．

■4096色のカラー表示が可能な「NeXTstation Color」

NeXTstation Colorは，NeXTstationのディスプレイ表示をカラー化したもの．1.5MbytesのVRAMを搭載し，4096色の同時表示が可能となっている．解像度1120×832ドットの16インチカラーディスプレイを採用した．

標準メインメモリが12Mbytesとなっているほかは，NeXTstationと同一スペック．

写真 2 4995ドルとワークステーションとしては，最も低価格帯に属する「NeXTstation」．

写真 3 NeXTdimensionの画面．左側のウィンドウには，NTSCの動画像が映し出されている．

■i860を搭載したフルカラーボード「NeXTdimension」

1670万色の同時表示可能な24bitカラーグラフィックボードNeXTdimensionは，NeXTcubeの拡張バススロットに接続して使用するもの．描画専用プロセッサとしてクロック周波数33MHzのi860を搭載し，30000ポリゴン／秒の描画速度を実現した．また，JPEGイメージ圧縮／展開用プロセッサとして，米C-Cube Microsystems社のCL550を搭載し，リアルタイムでの静止画像の圧縮／展開を可能にしている．

メインメモリは，標準で8Mbytesを搭載し，最大32Mbytesまで拡張が可能．ディスプレイ用メモリとして，4MbytesのVRAMを装備している．

そのほか，NTSCのビデオ信号のインプット／アウトプットを装備し，NeXTstepのウィンドウ内にビデオイメージを表示することもできる．

■添付のNeXTstepはRelease2.0に

上記製品の発表に合わせて，NeXTstepもRelease2.0にバージョンアップされた．機能強化された主な点は，①カラーへの対応，②アプリケーションビジーカーソルの追加，③Workspace Managerの改良，④FAX機能の追加，⑤DOSコンパチブルフロッピーのサポート，⑥CD-ROMドライブのサポート，⑦NeXTmailの機能強化――など．現行のRelease1.0とは，バイナリコンパチブルを実現している．

Release2.0は，StandardバージョンとExtendedバージョンの2種類が用意された．この2つはバンドルされるアプリケーションの内容に若干の差があり，105Mbytesハードディスク内蔵モデルにはStandardバージョンが，340Mbytes以上のハードディスク内蔵モデルにはExtendedバージョンが添付される．

また，米Lotus Development社，米Ashton-Tate社，米Adobe Systems社，米WordPerfect社，米Sybase社など50社以上が，NeXTstep2.0対応のアプリケーションソフトウェアの開発を行なうと発表している．

☆

価格は，NeXTcube（標準構成，含モノクロディスプレイ）が7995ドル，NeXTstation（同）が4995ドル，NeXTstation Color（標準構成，含カラーディスプレイ）が7995ドル，NeXTdimensionが3995ドル．現行ユーザーのマザーボードアップグレード料（含NeXTstep2.0）は1495ドル．販売開始は，NeXTcubeとNeXTstation が今年11月，NeXTstation ColorとNeXTdimensionが1991年第1四半期の予定．

なお，日本を含む極東地域ではキヤノン㈱が販売を行なう．価格は，NeXTcubeが178万円，NeXTstationが108万円，NeXTstation Color が178万円，NeXTdimensionが90万円．現行ユーザーのアップグレード料は29万円の予定．販売開始は，NeXTcubeとNeXTstationが1991年1月，NeXTstation ColorとNeXTdimension が1991年3月の予定．

写真 4 NeXT社Steven Jobs社長は，「すでに1万5000台を超えるオーダーを受注している」と語った．

問い合わせ先：〒211 川崎市幸区鹿島田890-12 新川崎三井ビル ☎044-549-5111

ソニーがRISC NEWSに新機種を追加

ソニー㈱は，CPUにRISCプロセッサを採用したワークステーションのRISC NEWSシリーズに「NWS-3870」を追加，販売を開始した．

CPU，FPUには，クロック周波数25MHzのR3000，R3010を採用し，25MIPSの能力を誇っている．メインメモリは64Mbytesを標準で搭載し，最大128Mbytesまで拡張可能．また，命令とデータキャッシュをそれぞれ64Kbytesずつ搭載している．

主な特徴として，①4MbytesのSIMMの採用，②容量1.25Gbytesのハードディスクを内蔵，③I／Oプロセッサとして68030を採用，④オーディオインターフェイスを装備――など．

また，CPUにRISCを採用し，I／Oプロセッサを搭載しない「NWS-3720」の販売も開始した．

CPU，FPUは，クロック周波数20MHzのR3000，R3010を採用し，17MIPSの能力を誇っている．メインメモリは16Mbytesを標準で搭載し，最大128Mbytesまで拡張可能．容量640Mbytesのハードディスクを内蔵する．

なお今回，OSもバージョンアップを行ない，NEWS-OS Release 4となった．同OSより，X Window System Ver.11 Release 4をサポートし，Motifの搭載も行なう．また，ユーザー管理などをメニューにより設定可能なNEWS-DESKTOPおよびネットワーク管理ツールのSNMPのサポートを行なう．

価格は未定となっている．

表1　NWS-3870の主な仕様

CPU	R3000（クロック周波数25MHz）
数値演算コプロセッサ	R3010（クロック周波数25MHz）
I／Oプロセッサ	68030（クロック周波数25MHz）
メモリ	64Mbytes（最大128Mbytes）
ハードディスク	1.25Gbytes内蔵
拡張スロット	3
OS	NEWS-OS Release4，NFS，X11R4，Motifなど

写真　NWS-3870．64Mbytesのメモリ，1.25Gbytesのハードディスクを内蔵した，ネットワークサーバ向けの上位モデルである．

オムロンが68040採用のLUNAを発売

オムロン㈱は，従来のLUNAシリーズと上位互換のワークステーション「LUNA-Ⅱ」を開発，販売を開始した．

同マシンは，CPUにクロック周波数25MHzの68040を採用し，20MIPSの処理能力を実現している．メインメモリは，標準で32Mbytesを搭載し，最大64Mbytesまで拡張可能である．

主な特徴として，①従来のLUNAで稼働している400種類以上のアプリケーションを利用可能，②2チャンネルのEthernetをサポートし，LANの高速化とゲートウェイ化を実現，③チラツキの少ない垂直周波数66Hzのディスプレイを採用，④テンポラリファイルの効率化を図ったRAMディスクを搭載，⑤拡張性が容易なブロック追加式のI／Oを採用，⑥市販のVMEボードが利用可能なVMEボックスを用意，⑦ASCII，JIS，ASCII＋JISの3種類のキーボードをサポート――など．

OSは，UniOS-MachとUniOS-Uより選択可能．また，X Window System Ver.11 Release 4をサポートし，グラフィカルユーザーインターフェイスのMotifに日本語処理ソフトWnnを付加した日本語Motifを搭載している．

出荷は来春を予定しており，価格は未定となっている．

表1　LUNA-Ⅱの主な仕様

CPU	68040（クロック周波数25MHz）
メモリ	32Mbytes（最大64Mbytes）
ハードディスク	250Mbytes内蔵
フロッピーディスク	3.5インチフロッピーディスク
拡張スロット	PC-9801互換スロット×2
サイズ	398（W）×365（D）×122（H）mm
OS	UniOS-MachまたはUniOS-U，X11R4 日本語Motif，Wnn4.0，NFSなど

問い合わせ先：オムロン㈱ EWSシステム事業部販売促進課
〒140 東京都品川区北品川4-7-35
御殿山森ビル
☎03-5488-3253

日本サンが日本語OpenWindowsを開発

日本サン・マイクロシステムズ㈱は，ウィンドウ環境のOpenWindowsを日本語化した「日本語OpenWindows」の販売を開始した．

日本語OpenWindowsは，グラフィカルユーザーインターフェイスのOPEN LOOKの日本語利用を実現した，Sunのウィンドウ環境である．

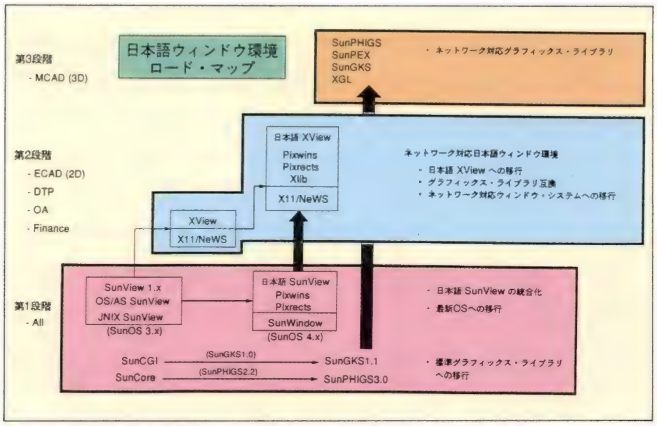
主な特徴として，①各国言語に対応する入力方式や表示方式，数値，日付のフォーマットなど固有なデータをアプリケーションのソースコードと分離し，設定する環境切り替え機能をサポート，②ANSI/C，POSIX，AT&T MNL3.2，XIMなどの国際標準仕様に準拠し，マルチバイトコードハンドリング，ワイドキャラクタなどをサポート，③日本語の入力とともに，画面コンポーネント，メニュー，ヘルプメッセージの日本語化などの日本語利用に必要な機能の装備，④システムフォントとしてゴシック体を採用，――など．

また，OpenWindowsのコンポーネントとして，OPEN LOOKの仕様に基づいたアプリケーション開発をサポートする，日本語XViewの提供を行なう．XViewを使用して開発された英語アプリケーションは，環境切替機能により，容易に日本語化が可能であるという．

日本語OpenWindows使用権は無料，日本語OpenWindowsの開発環境である日本語Devguide使用権は1万円である．ただし，どちらもメディアとマニュアルに関しては有料で7万円となっている．

問い合わせ先：〒102 東京都千代田区二番町11-19
興和二番町ビル
☎03-221-7021

日本語OpenWindowsでは，メニュー，ボタンなどのすべての日本語化が行なわれている．



現在は，図中の第1段階から第2段階への移行期であり，第1.5段階といえる．

富士通がメインフレーム用UNIXを発売

富士通㈱は，UNIX SystemV Release 4を採用したメインフレーム用のUNIX「UXP/M」を開発，同社のスーパーコンピュータVP2000シリーズおよび汎用コンピュータMシリーズ(M-780/770/760モデルグループ) 向けに販売を開始した．

主な特徴として，①アプリケーションのレスポンスタイムを改善するリアルタイム機能を提供，②SystemVやBSDなどの種類の異なったファイルシステムを1つのファイル中に実現するVFS(仮想ファイルシステム) を実現，③ファイルをメモリ上に展開することにより，大幅な処理性能の向上を可能とするメモリマップファイル機能を提供，④X/Open，POSIXに準拠，⑤国際化のための基本機能MNLS (Multi National Language Supplement) に準拠した日本語機能の提供，⑥X Window System Ver.11 Release 4とグラフィカルユーザーインターフェイスのOPEN LOOKを提供，⑦MシリーズのEXA (EXtended system Architecture) をサポートし，最大256チャンネル，2Gbytesのユーザー空間，I/Oスループットの向上など，大量処理に対応する機能を強化――など．

価格は，UXP/M 基本部で月額21万円より．

問い合わせ先：〒100 東京都千代田区丸の内1-6-1
丸の内センタービル システム第一販売推進部
☎03-216-3211

ソニーが386SX搭載のノート型パソコンを発表

ソニー㈱は，CPUに386SXを搭載し，ハードディスクを内蔵したノート型パーソナルコンピュータ「QL／Note」を開発，販売を開始した．

CPUの動作クロック周波数とメモリ，内蔵ハードディスクの容量により，PCX-310NR 3／5／7の3モデルを用意している．NR 3は，クロック周波数16MHzにメモリ1Mbytes，内蔵ハードディスク20Mbytes，NR 5は，クロック周波数16MHzにメモリ1Mbytes，内蔵ハードディスク40Mbytes，NR 7は，クロック周波数20MHzにメモリ2Mbytes，内蔵ハードディスク40Mbytesである．

主な特徴は，①全モデルに2.5インチのハードディスクを搭載，②JEIDA Ver. 4に対応したICカードインターフェイスを標準装備，③解像度640×480ドットの8階調表示が可能なバックライト付き液晶ディスプレイを採用，④RS-232C，プリンタ，オプションのテンキーのそれぞれのインターフェイスを装備，⑤Ni-Cd電池を内蔵し，1時間のバッテリ駆動が可能，⑥MS-DOS，MS-Windows，日本語入力FPのVJE-βなどをハードディスクにインストールして出荷，⑦キーボードにボールマウスを標準装備――などど．

サイズは315（W）×262.5（D）×45（H）mm，重さは3kg．

価格は，NR 3が39万8000円，NR 5が49万8000円，NR 7が59万8000円となっている．

フロッピーディスクを分離したデスクトップユニットを用意

QL／NOTEは，3.5インチフロッピーディスクなどを内蔵した合体型のデスクトップユニットを3タイプ用意している．

表1　主要スペック一覧表

	NR 3	NR 5	NR 7
CPU	80386SX		
クロック周波数	16MHz		20MHz
メインメモリ（最大）	1 Mbytes（5 Mbytes）		2 Mbytes（6 Mbytes）
ハードディスク	20Mbytes	40Mbytes	
ICカード	JEIDA Ver. 4		
インターフェイス	RS-232C，プリンタ，テンキー		
バッテリ駆動時間	約1時間		
ディスプレイ	JEGA640×480ドット		
サイズ	315（W）×262.5（D）×45（H）mm		
重量	約3.0kg		

タイプ1（8万円）は，電源，3.5インチフロッピーディスク，CRTインターフェイス，タイプ2（12万円）はタイプ1＋ハーフサイズの拡張スロット，タイプ3（22万円）はタイプ2＋Ethernetインターフェイスをそれぞれ内蔵している．

本体と同ユニットは，専用120ピンの拡張バスで接続を行なう．

写真1　キーボードの右奥には，ボールマウスが標準で装備されている．

写真2　本体にデスクトップユニットを装着した．デスクトップユニットの側面のレバーにより簡単に着脱が行なえる．

松下が業界最薄型の ノート型パソコンを発売

松下電器産業㈱は，CPUにクロック周波数16MHzの386SXを搭載したノート型パーソナルコンピュータ「Panacom PRO NOTE」を開発，販売を開始した．

フロッピーディスクと20Mbytesのハードディスク内蔵の２モデルがあり，主な特徴として，①Panacom Mシリーズとソフトウェア互換を実現，②EMS対応のメモリ２Mbytesを標準装備，最大４Mbytesまで拡張可能，③バックライト付きの16階調白黒液晶を採用，④レジューム機能を装備，⑤約1.5時間のバッテリ駆動が可能――など．

サイズは310（W）×254（D）×44（H）mm，重さはフロッピーディスクモデルが約2.9kg，ハードディスクモデルが3.1kg．

価格は，フロッピーディスクモデルが27万8000円，ハードディスクモデルが43万8000円となっている．

薄型ながらオプションの2400bpsMNPクラス５のモデムも内蔵可能である．

沖電気がAX仕様の ノート型パソコンを発売

沖電気工業㈱は，CPUにクロック周波数16MHzの386SXを搭載したノート型パーソナルコンピュータ「if NOTE」を開発，販売を開始した．

同マシンには，ディスプレイの有無，3.5インチフロッピーディスクまたは20Mbytesの内蔵ハードディスクの違いにより，４モデルを用意．主な特徴として，①フロッピーディスクモデルはメインメモリ１MbytesにRAMディスク１Mbytesを，ハードディスクモデルはメインメモリ１Mbytesを標準装備，②解像度640×480ドット，バックライト付き８階調のペーパーホワイト液晶ディスプレイを採用（ディスプレイ付属モデル），③インターフェイスとして，RS-232C，プリンタ，拡張フロッピーディスク，CRT，テンキー，拡張ユニットを標準装備，④レジューム機能を装備，⑤1.2時間のバッテリ駆動が可能，⑥オプションの2400bpsMNPクラス５のモデムを内蔵可能（ただし，バッテリ駆動は不可），⑦オプションの一体型拡張ユニットとの合体により，通信をはじめとしたシステム拡張が可能――など．

日本語／英語モードに対応したMS-DOS Ver.3.21を標準装備している．

サイズは315（W）×249（D）×48（H）mm，重さ約３kg．

価格は，フロッピーディスクモデルでディスプレイなしのモデル20が23万8000円，液晶ディスプレイ付きのモデル20Lが29万8000円，ハードディスクモデルでディスプレイなしのモデル21が33万8000円，液晶ディスプレイ付きのモデル21Lが39万8000円，オプションの１Mbytes増設メモリボードが４万8000円，２Mbytes増設メモリボードが７万8000円となっている．

表１　拡張ユニットの仕様

ＡＨ	ＡＸハーフサイズ拡張スロット×２，20Mbytesハードディスク	23万8000円
ＡＦ	ＡＸハーフサイズ拡張スロット×２，3.5インチフロッピーディスク	14万8000円
ＴＨ	ＡＴフルサイズ拡張スロット×１，20Mbytesハードディスク	23万8000円
ＴＦ	ＡＴフルサイズ拡張スロット×１，3.5インチフロッピーディスク	14万8000円

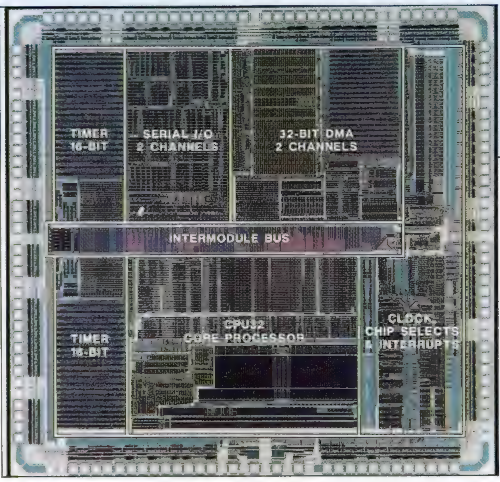
液晶ディスプレイは取り外しが可能で，オプションの反射型液晶ディスプレイの接続も可能である．

モトローラが制御用32bitプロセッサを発表

日本モトローラ㈱は，32bitCPUコアユニットを使用したMC68300ファミリの新製品「MC68340」を発表した．同LSIは，MC68000をベースに大ブロックの高速データ転送能力を必要とする応用分野向けに設計され，CD-IプレーヤやLBPの制御などに向くもの．

同LSIは，CPU32と呼ぶMC68020をベースとしたコアプロセッシングユニットに，オンチップの周辺装置を搭載している．CPU32コアは，MC68000およびMC68010と完全な互換性を持ち，さらにMC68020の持つ命令の32bit拡張が追加されている．処理速度はMC68000の3倍で，約2MIPS．

オンチップ周辺装置の中で最も特徴的なのは，32bitデュアルチャンネルDMAコントローラで，これはCPU32の介在なしにハードディスクとメモリの間で最大33Mbytes／秒のデータ転送を可能にする．また，2つのシリアルI／Oチャンネルは，それぞれ独立した全2重の同期／非同期のレシーバ／トランスミッタを持ち，端末やホストとの通信の際には最大3Mbytes／秒の転送速度を実現する．



MC68020をベースとしたコアプロセッシングユニットに，オンチップの周辺装置を搭載した「MC68340」．

動作クロック周波数は16MHzで，サンプル供給は1990年第4四半期後半，サンプル価格は1万円の予定．量産供給開始は1991年第1四半期中の予定．また，評価用のシステムボード「M68340EVS」も1990年第4四半期後半より供給される．同ボードの価格は15万円の予定．

モトローラが高速SRAMの新製品を発表

日本モトローラ㈱は，汎用1MbitSRAMの新製品を2種，特定用途向け専用SRAMの新製品を2種発表した．

汎用1MbitSRAMは「MCM6226」(128K×8bit)と「MCM6228」(256K×4bit)で，ともに0.8μCMOS技術で設計・製造され，アクセス時間25nsと30nsの2種類を用意した．パッケージは，MCM6226が32ピンのSOJ，MCM6228が28ピンのSOJである．

特定用途向け専用品は，16K×16bit構成のシンクロナスRAM「MCM62990」とラッチ付きRAM「MCM62995」．それぞれ1.0μCMOS技術で設計・製造され，52ピンのPLCCパッケージで供給される．両製品は，マイクロプロセッサに対してコンパチブルで高性能なキャッシュメモリを提供するために設計されたもの．MCM62990は，完全な同期式であり，40MHzまでのクロック周波数で汎用キャッシュメモリとして使用可能．MCM62995はラッチを持つ非同期デバイス．アクセスタイムはともに17ns．

サンプル価格は，MCM6226とMCM6228の25nsのものがともに1万5000円の予定，MCM62990が8610円，MCM62995が9490円．

富士通がアウトラインフォント処理用LSIを発売

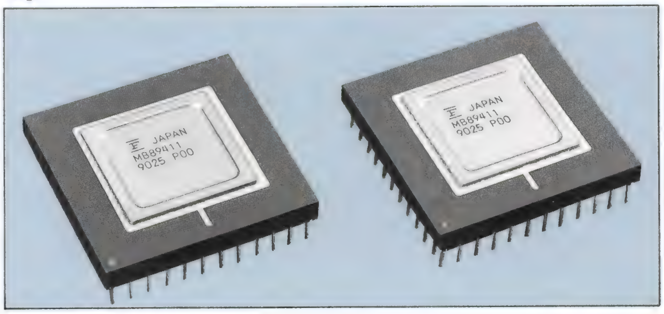
富士通㈱は，アウトラインフォントなどの処理を高速に行なう画像処理用LSIとして，キュービックカーブエンジン「MB89411」を開発，発売した．

キュービックカーブエンジンとは，曲線を定義する座標列を与えることによって，発生する3次曲線を直線で近似し，その座標列を出力する処理を行なう専用LSIである．同LSIは，3次ベジェ曲線・3次Bスプライン曲線の直線列変換と，曲線形式変換の機能を持ち，クロック周波数20MHzで動作した場合の処理速度は最大100万座標／秒．

主な特徴は，①近似精度は任意に設定可能で，PostScriptのフラットネス指定にも対応可能，②独立した入出力バスを持ち，非同期パイプライン接続が可能――など．

サンプル価格は1万3000円．

アウトラインフォントなどの処理を高速に行なえるキュービックカーブエンジン「MB89411」．



IBMが新世代システムを発表

日本アイ・ビー・エム㈱は，汎用コンピュータ「IBMシステム／390（S390）」を発表した．

同システムは，①機器間の接続性を強化したシステム体系「ESA／390体系」，②「ES／9000プロセッサー・ファミリー」18モデル，③ESAを基に統合されたオペレーティングシステムファミリ「MVS／ESA 第4版」，「VM／ESA」，「VSE／ESA」，④8分野の新しい企業情報システム機能――で構成されている．

従来のESAをファミリ化して拡張

ESA／390システム体系は，従来のESAのハードウェアとソフトウェアをファミリ化し，それぞれの機能の拡張を行なった．機器間を高速に接続可能なESCONアーキテクチャ，ならびにセキュリティ管理を実現する暗号メカニズムを具体化したものである．

同システム体系の主な特徴は，①ESCONアーキテクチャに基づいて開発したESCONチャンネルより，光ファイバを利用し，ESCONディレクターで中継することで，最大10Mbytes／秒の転送速度で9kmまで離れた場所の入出力制御装置と接続が可能，②暗号メカニズムは，ES／9000の水冷プロセッサに暗号化機能を付加することによって，機密管理を必要とする情報を送信側で自動的に暗号化し，受信側で自動的に解読が可能．また，ウイルス対策として，決められた人間以外のアクセス防止機能のサポート，③従来のESA機能の拡張として，チャンネル間結合で接続した複数プロセッサの一元管理を実現したSysplexをサポートし，複数のコンピュータ間で稼働するMVSオペレーティングシステムを，1つのシステムとして，1台のターミナルから複数のシステムを操作，管理することが可能――など．

中型から大型汎用機まで18モデルを用意

ES／9000プロセッサー・ファミリーは，空冷式ラック型モデル4機種，空冷式フレーム型モデル6機種，水冷式フレーム型モデル8機種の計18モデルで構成している．

同ファミリーの主な特徴は，①1台のプロセッサを論理的に分割し，複数のオペレーティングシステムとして稼働させる「プロセッサー資源システム管理機構（PR／SM）」を全モデルに標準装備，②空冷式ラック型モデルは，電源の障害にも対処可能なバッテリ付きバックアップ機構を標準装備，③水冷式フレームモデルの上位2モデルは，TCM（熱伝導モジュール）と複数命令実行機構の装備，高速緩衝記憶機構の二重化などにより，大幅に処理能力を向上した．

従来の中大型システム用の3つのOSを統合

MVS／ESA（多重仮想記憶／エンタープライズ・システム体系）は，従来の機能を拡張するとともに新機能を追加した，MVS／ESA第4版となった．また，従来のVM／SP，VM／HPO，VM／XAの3つのオペレーティングシステムをVM／ESA（仮想計算機／エンタープライズ・システム体系）として統合し，データ空間を拡大するなど機能の拡張を行なった．VSE／ESA（拡張仮想記憶／エンタープライズ・システム体系）は，利用可能な記憶域の大きさを拡大し，MVS／ESAとの親和性を強化している．

システム支援機能を強化

企業情報システム機能の特徴として，①OSIとSNAとの接続，TCP／IPによる通信が可能，②CICS，IMSといったオンライン処理サブシステムの管理機能を強化し，処理効率を向上，③SAAに基づく，企業内のすべてのシステム管理の統合化，自動化を目指したSystemViewのサポート，④高速なデータ暗号化機能のサポート，⑤ユーザーレベルに合わせたセキュリティの実現，⑥空冷式フレーム型モデルにもベクトル機能を装備し，技術計算機能を実現――など．

価格は1800万円から

概算価格は，最も安い9221プロセッサーモデル120が1800万円から，最も高い9021プロセッサーモデル900が55億2900万円からとなっている．MVS／ESA第4版の月額利用料金は12万1100円からである．

そのほか，9032ESCONディレクターは990万円から，9033ESCONディレクターは2970万円，VM／ESA第1版（日本語版）は月額8万5600円から――などとなっている．

空冷式ラック型，空冷式フレーム型，水冷式フレーム型の3タイプがあり，合計18機種を用意している

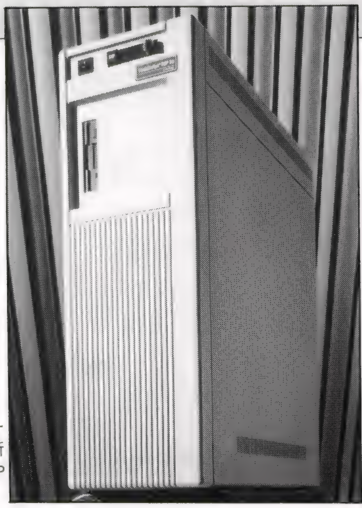
Linotypeが日本語PostScriptRIPなど3製品を発表

英Linotype社は，カラーに重点をおいた新製品3機種を開発したと発表した．内訳は，①最新の日本語PostScriptRIP「RIP30-J」，②新型ラスタイメージプロセッサ「Linotronic230」③600dpi高品質プリンタ「Printer60-J」――の3製品．同社は，昨年9月に初の日本語PostScriptRIPおよびイメージセッタを発表したが，今回の発表で日本語PostScriptに強力な製品ラインが構成される．

RIP30-Jは，カラー，文字，画像すべてをサポートし，従来のプロセッサに比べて3倍のスピードを実現，現在市場に出ているPostScriptRIP中で最高速である．同社が独自に開発したネプチューンテクノロジーにより，特にカラーハーフトーン処理にすぐれた威力を発揮する．CPUにクロック周波数33MHzの68030を採用．Ethernetの内蔵により，高速ネットワーキングを実現している．

Linotronic230は，RIP30-JおよびRIP3-Jと組み合わせて，文字，画像の統合ページをカラー出力する．1インチ当たり133線のハーフトーンを可能にし，解像度は1693dpiから635dpiまで4段階が設定可能で，対応する最高印刷速度は15から40cm／分．最大出力幅は305mm．CPUには68008を採用している．

Printer60-Jは，同社が初めて出す普通紙レーザプリンタ．解像度600dpiの高品質日本語PostScript対応で，最大A3サイズを出力できる．最大速度12枚／分（A4），標準でEthernetを内蔵．フォントは，リュウミンとゴシックの2書体を搭載．CPUには33MHzの68030を採用し，Linotype独自のネプチューンRIPテクノロジーを取り入れて，プリンタとして最高性能を実現している．



カラーに最適のネプチューンテクノロジーによる最新型日本語PostScriptRIP「RIP30－J」

価格は，RIP30-Jが750万円，Linotronic230が520万円，Printer60-Jが800万円．

問い合わせ先：〒102 東京都千代田区二番町3 白揚ビル
ライノタイプ東京広報室
☎03-263-5985

インテルとアプリックスがPC-9800用DVI開発プラットフォームなどを発表

インテル㈱と㈱アプリックスは，PC-9800上でDVI技術を利用するためのハードウェア「アクションメディア98」を共同開発，発売した．また，㈱アプリックスは，世界初のスタンドアローンDVIプレーヤ「デジタルポップ」を独自に開発したと発表した．

アクションメディア98は，CD-ROMドライブ，DVI再生ボード，スピーカを組み込んだ外部ユニットおよびPC-9800とそのユニットを接続するためのPC-9800用拡張カードとケーブルの組み合わせで構成される．外部ユニットにモニタを接続することにより，PC-9800がフルスクリーン動画／静止画／オーディオの再生のできるマルチメディアプラットフォームマシンとなる．また，DVIソフトウェア開発用に，キャプチャーボード，プロダクションツール，ソフトウェアライブラリを用意しており，これを利用することで，DVIアプリケーションオーサリングツールを構成できる．

世界初のスタンドアローンDVIプレーヤ「デジタルポップ」．CD-ROMに記録したDVIアプリケーションを実行／制御可能．

デジタルポップは，CD-ROMドライブを装備した本体，リモコンまたはタッチスクリーンを装備したディスプレイで構成され，あらかじめCD-ROMに記録したDVIアプリケーションを実行／制御できる．店頭用ディスプレイなどをターゲットとしており，これらソフトウェアの開発／制作に対しては，同社および関係数社で対応するとしている．

価格は，アクションメディア98が69万8,000円，デジタルポップが98万円．

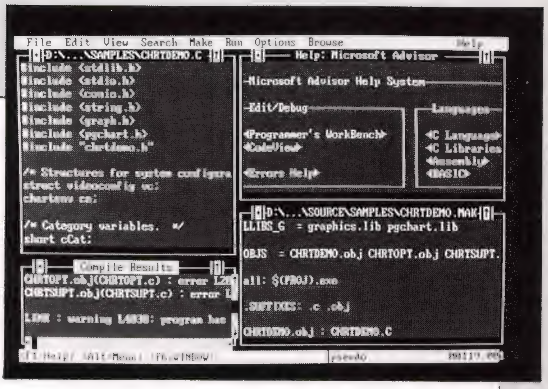
問い合わせ先：〒103 東京都中央区日本橋小網町4-4
㈱アプリックス
☎03-808-2064

マイクロソフトがMS C6.0の日本語版を発表

マイクロソフト㈱は，プログラミング言語「Microsoft C Professional Development System Version6.0」を日本語化し，11月16日より発売すると発表した．

同ソフトは，1988年に発売されたVer.5.1からのバージョンアップであり，主な改良点は，①ユーザーによる機能拡張も可能な統合開発環境「プログラマーズ・ワークベンチ」を装備，②関数全体に対するグローバルオプティマイズ機能の付加，③インラインアセンブラの統合化など，新しい言語仕様を採用，④ソースレベルデバッガ「コードビュー」の機能強化，⑤OS/2対応機能をサポート――など．

対応機種は，80286／80386CPUを搭載したPC-9800シリーズおよびPC-98シリーズのノーマルモード．動作環境は，MS-DOS Ver.3.1以上とメインメモリ640Kbytes以上，またはOS/2 Ver.1.1以上とメインメモリ1Mbytes以上，ハードディスクの空き容量6Mbytes以上が必要．



新開発環境"プログラマーズ・ワークベンチ"やデバッガ"コードビュー"を装備した「Microsoft C Professional Development System Version6.0」．

価格は9万8000円．現行ユーザーのバージョンアップ料は，Ver.5.1からが2万5000円，Ver.4.0以前からが4万円．

なお，同ソフトの発売にあわせてサードパーティ20社より同ソフトに対応したライブラリパッケージ51種が発売される．

問い合わせ先：〒160 東京都新宿区西新宿7-5-25 Kビルディング
☎03-363-1200

リード・レックスがWindows対応のDTPソフトを発売

㈱リード・レックスはPC-9800シリーズ用の日本語DTPソフトウェア「PRESBOX Windows版」を開発，発売した．

主な特徴は，①文字入力，編集，図形描画，作表，レイアウトなどの機能を備えており，ワープロソフトに似た操作性で使用可能，②MS-Windowsを利用したWYSIWYGインターフェイスを採用，③高度な日本語編集能力の装備とともに，和欧文混在文書にも対応，④各種ソフトウェアからのデータ取り込みをサポート，⑤日本語PostScriptを始め，各種ページプリンタのネイティブモードをサポート――など．

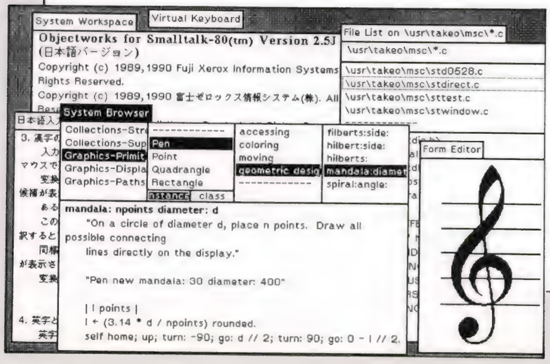
対応機種は，80286／80386CPU搭載のPC-9800シリーズで，ハイレゾモードもサポートしている．動作環境は，MS-DOS Ver.3.3とMS-Windows Ver.2.11またはMS-Windows／386 Ver.2.11，メインメモリ640KbytesとEMSメモリ1Mbytes以上が必要．また，ハードディスクが必須となっている．

価格は9万8000円．

問い合わせ先：〒150 東京都渋谷区渋谷1-9-8 渋谷朝日野村ビル
☎03-797-3911

富士ゼロックス情報システムがPC-9800用Smalltalk-80を発売

富士ゼロックス情報システム㈱は，PC-9800シリーズ用のオブジェクト指向型ソフトウェア開発環境「Smalltalk-80システム・パッケージ」を開発，販売を開始した．



PC-9800シリーズ上でオブジェクト指向プログラミングを可能にする「Smalltalk-80システム・パッケージ」．

今回発売したのは，英語版のV2.5と日本語機能付きのV2.5J．同ソフトウェアは，クラス階層構造／インへリタンス機能，プログラムのモジュール化，データの隠ぺい／抽象化，動的結合などのオブジェクト指向プログラミングを実現しており，再使用可能な300以上のクラス定義と数千のメソッド定義で構成されたクラスライブラリを装備している．

対応機種は，80386を搭載したPC-9800シリーズ．動作環境は，MS-DOS Ver.3.3以上，プロテクトモードメモリがV2.5は640Kbytes＋3Mbytes以上，V2.5Jは640Kbytes＋4Mbytes以上，ハードディスク空き容量がV2.5は10Mbytes以上，V2.5Jは15Mbytes以上必要．

価格は，V2.5が12万円，V2.5Jが14万円．

問い合わせ先：〒160 東京都新宿区西新宿3-16-6
☎03-378-8011

オムロンが日本語Motifを発売

オムロン㈱は，グラフィカルユーザーインターフェイス「OSF／Motif」を日本語化し，同社のマルチRISCワークステーションLUNA-88K用のOSであるMachにバンドリングして発売した．

OSF／Motifは，OSF (Open Software Foundation) が提唱するGUIで，①X Window上で動作，②WnnとXwnmoを使用して日本語入力が可能，③OS/２のPresentation Managerと動作互換を実現，④アイコンやウィンドウなどの３次元表示を採用——などの特徴を持つ．

日本語MotifがバンドリングされたMach OSの価格は22万円．

LUNA-88KのOSであるMachにバンドリングされるGUI「日本語OSF／Motif」．

問い合わせ先：〒140 東京都品川区北品川4-7-35 御殿山森ビル
☎03-5488-3253

CSKがUNIX用のリアルタイム・スプレッドシートを発売

㈱CSKは，米UniPress Software社が開発したUNIXワークステーション用リアルタイム・スプレッドシートソフトウェア「Q-Calc Realtime」の日本国内での独占販売権を獲得したと発表した．CSKが日本語化を行ない，11月より販売を開始，1991年１月から出荷を開始する予定．

リアルタイム・スプレッドシートとは，時々刻々変化するデータをある時間間隔で取得し，自動更新・自動再計算・自動再表示を行なう表計算ソフトウェアで，金融機関のディーリング／トレーディング業務などに利用されている．

同ソフトの主な特徴は，①Lotus1-2-3に似たユーザーインターフェイスを採用，②X Windowに対応，③データ取得の時間間隔は10秒／30秒／60秒／５分のいずれかからユーザーが指定可能，④オプションでSybaseデータベースシステムとのインターフェイスをとることも可能——など．

対応機種は，RS6000，DECstation，SPARCstation，HP9000，Apollo Domain，NCR Towerなど．

価格はワークステーション用が38万円から，ミニコン用が85万円からを予定．

問い合わせ先：〒163 東京都新宿区西新宿2-6-1 新宿住友ビル
☎03-344-1811

アステックがSaber-Cの日本総販売代理店に

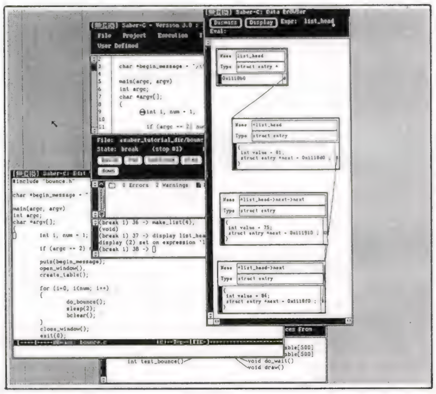
㈱アステックは，米Saber Software社が開発したUNIXワークステーション用プログラム開発ツール「Saber-C」の日本総販売代理店として契約，同ソフトの日本語環境を開発，発売した．

同ソフトは，UNIX上でのC言語によるプログラムの開発から保守までの工程をサポートするもので，インクリメンタル・リンカ，静的／動的エラー検出機能，ソースレベルデバッガ，ブラウザを１つのシステムにまとめたもの．主な特徴は，①X Window，SunViewなどのウィンドウ環境を活かしたグラフィカルユーザーインターフェイスを採用，②内蔵インタプリタによるプロトタイピングが可能，③グラフィックブラウザにより関数呼び出しやデータ構造の視覚化が可能——など．

対応機種は，Sun３／SPARCシリーズ，DECstation，VAXstation．年内には，IBM，HP，Data Generalの各社のワークステーションにも対応する予定．動作環境としては，最低８Mbytesのメモリが必要．

価格は，１年間のサポート料込みで70万円．

問い合わせ先：〒150東京都渋谷区道玄坂1-22-12長島第一ビル
☎03-477-1541



UNIX上でのC言語によるプログラムの開発から保守までの工程をサポートするプログラム開発ツール「Saber-C」．

エルゴソフトとYHPがEGWordをHP9000シリーズに移植

㈱エルゴソフトと横河・ヒューレット・パッカード㈱は，Macintosh上のワープロソフトであるEGWordをHP9000シリーズのOSF／Motif上に移植，YHPから「HP EGWord Ver.3.1」として販売を開始した．

同ソフトは，HP9000シリーズ300／800，HP Apollo9000シリーズ400のHP-UX上のOSF／Motifで使用可能．対応プリンタは，HP-UX対応の日本語レーザープリンタ（YSシリーズ）とリコーのLP3320-SP 4である．日本語入力処理には，EGBridgeを使用し，文書フォーマットのEDF（Extended Document Format）を利用することにより，Macintosh版とのデータ互換性を実現している．

YHPは，日本語入力FPとして「HP EGConvert」の販売も開始した．同FPは，エルゴソフトのEGConvertをHP-UXのプログラムインターフェイスOpenNLIO／Xを用いて，X Window用のフロントエンドプロセッサとしたもので，CASEツールのSoftBench，日本語Teamworkなどの各種アプリケーションの日本語入力に利用可能である．

価格は，HP EGWord Ver.3.1が9万5000円，HP EGConvertが3万円となっている．

マルチフォント機能により明朝体やゴシック体など，文字の種類や大きさを自由に選択が可能である．

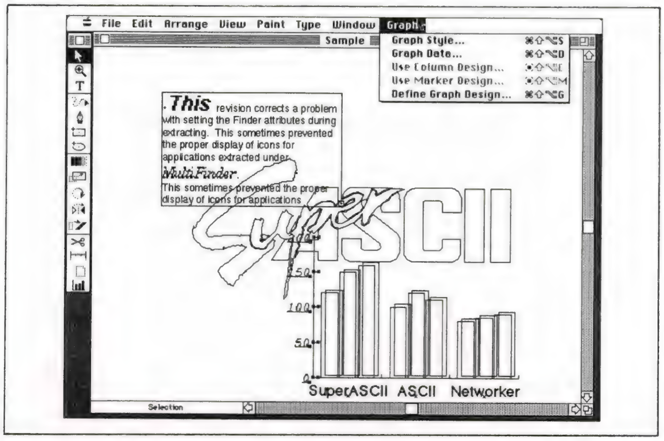
問い合わせ先：横河・ヒューレット・パッカード㈱
〒168 東京都杉並区高井戸東3-29-21
☎03-331-6111

アドビがIllustratorをバージョンアップ

アドビ システムズ ジャパンは，Macintosh用のイラストレーションソフトの新バージョン「Adobe Illustrator 3.0」英語版を，㈱システムソフトを通じて販売を開始した．

主なバージョンアップ点は，①テキストを直接画面上で入力，編集が可能，②テキストの長さ，文字スタイル，サイズおよびカラーが無制限に設定できるテキストのブロックが作成可能，③文字を曲線のパス上に配置でき，パス上を自由に移動可能，④PostScriptフォントを編集可能なアウトラインに変換し，独自のロゴや特殊な文字を作成可能，⑤テキストブロックに対して自由な角度での回転，変形や縮小，拡大が行なえ，変更後のテキストの直接編集が可能，⑥ワープロソフトからのテキストの読み込みが可能，⑦カーニング，行間，単語および文字間隔自動調整機能のサポート，⑧グラフ作成ツールをサポートし，棒グラフや帯グラフ，折れ線グラフなどのグラフをIllustrator 3.0上で作成可能，⑨ルーラー（定規）からガイドラインの引き出しが可能，⑩標準添付のAdobe Separatorの強化，⑪Adobe TypeManager2.0を標準添付――など．

グラフ作成機能により，ビジネスプレゼンテーション用の資料などもIllustratorだけで作成可能になった．



価格は10万円．なお現在，日本語化が進められており，予価は12万円，発売時期は未定．

Adobe Illustrator（英語版，日本語版）を，9月14日までに購入したユーザーは2万円，9月15日以降に購入したユーザーは5000円で3.0にアップグレードが行なえる．

問い合わせ先：㈱システムソフト 営業企画課
〒810 福岡県福岡市中央区天神3-10-30
☎092-722-4857

リコーがネットワーク用サーバを発売

リコー㈱は，CPUにクロック周波数16，25，33MHzの80386を採用しネットワークのサーバやワークステーションとして使用可能な「6386WGS」3機種を発売した．主な特徴は，①OSにMS-DOSとUNIX SystemVをサポート，②UNIXのネットワーク機能により，STARLANで接続したパソコンのネットワークサーバとして使用可能，③複数のプリンタを接続しプリンタサーバとして使用可能，④STARLANで接続されたパソコンを遠隔地のパソコンと公衆回線で接続したり，各種ネットワークと接続するためのゲートウェイとして使用可能，⑤IBM PC／AT互換アーキテクチャを採用しIBM PCの周辺機器とソフトウェアが使用可能――など．価格は81万円から．

日本ボードコンピュータが画像データベースを発売

日本ボードコンピュータ㈱は，PC-9800を使用して文字情報と画像情報を管理するリレーショナルデータベースシステム「PIXEL BASE Amie」を発売した．同システムは，画像圧縮ボード，フレームメモリ，データベースソフトで構成される．主な特徴は，①ホストとしてPC-9800を使用し基本システムを安価に構築可能，②画像情報は1670万色で表示し，1画面で最大3つの画像を同時表示可能，③画像データは平均12分の1に圧縮可能，④5万件のデータより1件を1秒以内に検索する高速検索を実現――など．システム価格は80万円．

NCDがX端末を発売

㈱日本NCD（☎03-5276-1756）は，19インチのX Window専用端末「NCD19b」を発売するとともに，X端末用のソフトウェア「NCDware2.2」をバージョンアップして発売した．NCD19bの主な特徴は，①解像度1024×800ドットのディスプレイを採用，②CPUにクロック周波数16MHzの68000を採用，④メモリは2Mbytesを標準装備し，5Mbytesまで拡張可能，③X11R4互換のXサーバを装備――など．NCDware2.2の機能強化した主な点は，①SNMP（Single Network Managenent Protocol）機能を装備，②X11R4との互換性を実現，③フォントキャッシング機能を装備，④XRemote機能のDEC環境への対応――など．価格は，NCD19bが39万8000円から，NCDware2.2が1サイト当たり9万8000円．

日立がISDN対応の光LANを発売

㈱日立製作所は，広帯域ISDNに対応した光LAN「Super LAN Σ-600」を発売した．主な特徴は，①620Mbpsのループ伝送速度を実現し，CCITTで標準化が行なわれたデジタルハイアラーキに準拠，②広帯域ISDNで用いられる予定のATMセルやATMセルの多重化を行なうSDH STMフレーム形式に準拠，③サポートインターフェイスとして，高速LANの国際標準規格であるFDDIインターフェイス，PBX／専用線インターフェイスの端末接続用のTTC標準2Mbpsデジタルインターフェイスおよび動画像装置接続用のNTSCインターフェイスなどを用意，④ループ交換機能やループバック機能をサポートし，ネットワーク全体を停止することなくノードの増設および移設が可能――など．価格は2ノード構成で6600万円から．

Appleが東京証券取引所外国部に上場

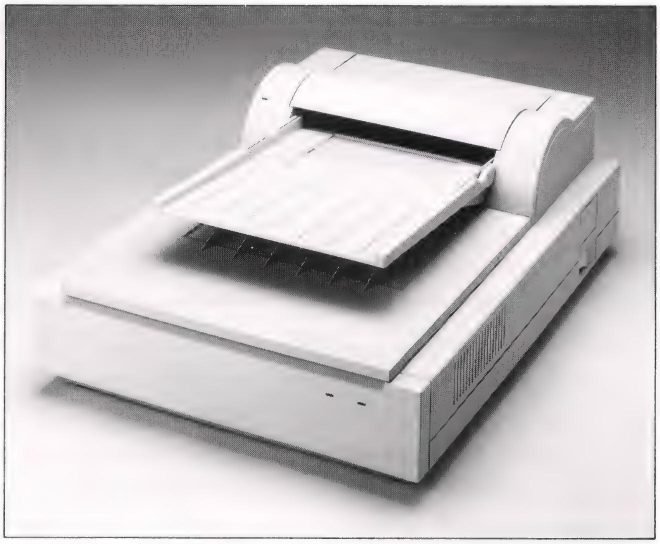
米Apple Computer社は，東京証券取引所外国部に上場したと発表した．同社会長兼最高責任者のJohn Scully氏は，「これはAppleと日本市場との末永い関わり合いを表わすものであり，日本という戦略上重要な市場に対するAppleの土台を今まで以上に強固なものとしていくと考える．将来日本がビジネス展開の中で大きな拠点となり，Appleの成長に欠くことのできない市場になると確信している」と語っている．

アクエリアスがテキストエディタを発売

アクエリアス・ソフトウェア開発㈲（☎0425-86-1031）は，テキストエディタ「シルヴィ」を発売した．主な特徴は，①ファイルにファイル名とは別の文書名を付けた管理が可能，②文書名を付けたファイルをフォルダ（書類入れ）に入れて分類，整理可能，③右寄せ，左寄せ，センタリングなどのレイアウト機能を装備，④マーク・アンド・ジャンプなどのプログラミング用のエディタとしての機能を装備，⑤印刷プログラム，文書整形プログラムなどを付属――など．対応機種はJ-3100シリーズ，PC-9800シリーズ．動作環境はメモリ384Kbytes以上，MS-DOS Ver.3.1以上が必要．価格は9000円．

リコーがワークステーション用のイメージスキャナを発売

リコー㈱は，UNIXワークステーションのイメージスキャナ「IS50／UX」の販売を開始した．主な特徴は，①SCSIインターフェイスを採用，②読み取り速度はA４判を240dpiで約５秒の読み取りが可能，③解像度は400dpiで256階調の読み取りが可能，④反転，ミラーイメージ，10種類のディザパターン，輝度調整，コントラストなどの機能を装備，⑤デバイスドライバと各種アプリケーションソフトの開発を容易にするスキャナ操作用ライブラリを用意，⑥サンプルプログラムとして，画像読み取りユーティリティを用意――など．対応機種は，SunシリーズとNEWSシリーズ．価格はSun用が52万8000円，NEWS用が51万8000円．



富士ゼロックスが小型・低価格のプリンタを発売

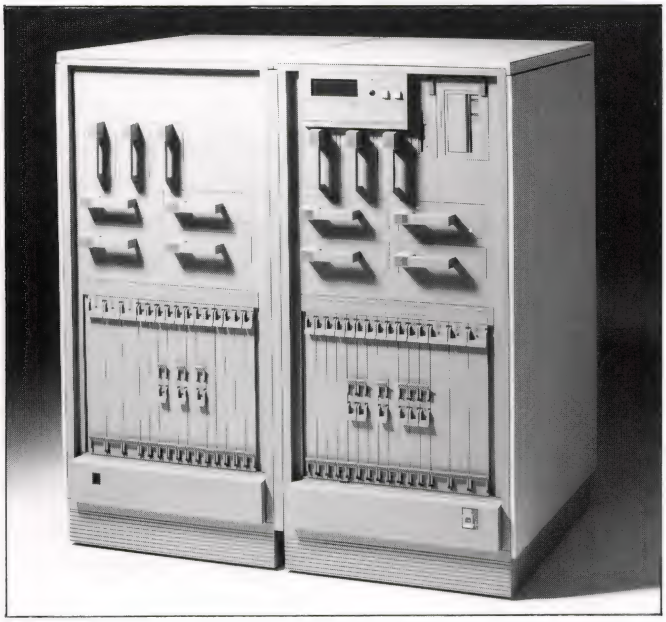
富士ゼロックス㈱は，コピー機能を備えたパーソナルコンピュータ用のプリンタを発売した．主な特徴は，①パーソナルコンピュータ用のプリンタとしての機能とコピー機能を装備，②305（W）×129（D）×201（H）mmの小型化を実現，③熱転写方式を採用し，プリント時48dB，コピー時43dBの低騒音を実現，④プリントスピードは約40秒（A４），コピースピードは約23秒（A４）を実現，⑤32×32ドットの文字フォントを搭載，⑥主走査８ドット／mm，副走査15.4／mmの解像度（コピー時）を実現，⑦３段階の濃度調整とハーフトーン機能により写真やイラストなどの濃淡を再現可能――など．対応機種は，J-3100／3300シリーズ，PC-9800シリーズ．価格は11万9800円．

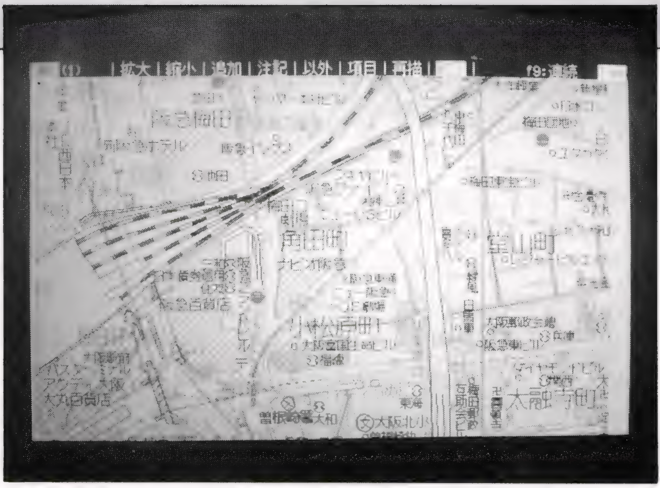
リコーがノート型日本語ワープロを発売

㈱リコーは，省スペース性を重視したノート型の日本語ワードプロセッサ「NL-２Ai」および「NL-３Ai」の２機種を発売した．主な特徴は，①NL-３Aiは内蔵電池で使用可能なバックライト付き640×400ドット白黒液晶画面を採用（NL-２Aiは反射型液晶），②約15万のAI用例に基づいて，構文意味解析手法によって変換を行なうAI自動かな漢字変換機能を搭載，③本体サイズは320（W）×250（D）×52（H）mm，本体重量は約2.6kgと小型軽量化を実現，④プリンタの分離が可能，⑤２つのCPUと３つのモーターを採用し印刷時間の短縮化を実現，⑥アウトライン処理方式を採用し，なめらかな拡大文字を表示可能，⑦内蔵のMS-DOSファイル変換ソフトによりパーソナルコンピュータとの文書データ交換が可能，⑧内蔵の電子手帳ICカードデータ変換ソフトによりシャープ製の電子システム手帳とのデータ交換が可能――など．価格はNL-３Aiが18万8000円，NL-２Aiが15万8000円．

富士通が高信頼の通信プロセッサを発売

富士通㈱は，ハードウェアだけでなくソフトウェアも分散・二重化したアーキテクチャを採用し，コンピュータネットワークの無停止・連続運転を可能にした通信処理プロセッサ「SURE SYSTEM 2000」を発売した．主な特徴は，①CPUとメモリを一体化したプロセッサモジュールを採用し，プロセッサモジュールは2～6台まで拡張可能で，並列分散処理が可能，②各プロセッサモジュールからの引き継ぎ情報を格納することにより，どのプロセッサからでも業務を高速に引き継げる共有メモリ方式を採用，③どのプロセッサモジュールからもディスクなどの周辺装置に直接かつ自由にアクセス可能で，I／Oの負荷や配置を自由に変えられるI／O直接共有アクセス方式を採用，④徹底したモジュール化実装方式により，システムの運転中でもモジュールの交換・増設が可能，⑤停電時には，データ内容を自動的にディスクにバックアップし，復旧時にはすぐに業務を再開可能なバッテリ・バックアップ方式を採用，⑥同一のOS機能を別々のプロセッサモジュールに配置し，ハードウェア障害やOS障害に対する耐故障性を実現し，システムを止めずにハードウェアの増設・保守が可能，⑦新旧複数世代を管理・動作できる機構やシステムの各種環境定義を運転中に変更できる機構を装備⑧ネットワークは，同社のFNA，OSI，TCP／IP，IBMのSNA，全銀協，JCAなど業界標準をサポート――など．レンタル価格は1カ月135万円から．





アルプス社が地図帳作成システムを開発

㈱アルプス社（☎052-762-2001）は，コンピュータマッピングによる地図帳作成システム「ALPS」を開発した．主な特徴は，①地図データの入力から編集・出力までの総合的な地図情報処理が可能，②ディスプレイ上で編集したデータは製版フィルムとして各色1枚ずつ出力され，従来の製図・製版作業が不要，③文字の書体・大きさ，線の種類・太さなどを簡単に変更でき，ぼかし表現などが可能，④地図の基本となる道路・河川・鉄道・等高線などのデータには，国土庁地理院発行の2万5000分の1地形図を基にデータ化し，大都会の中心部では1万分の1地形図の情報をデータ化したものを装備．⑤国土庁地理院の地形図には記載のない地図出版に必要な各種の情報は独自の調査によりデータ化したものを装備――など．同社では，このシステムを活用して，地図およびデータベースの作成の受注・DTPソフトの販売・データ出力サービスなどの事業を展開する．その第1弾として，すでに道路地図「ATRAS RD 近畿版」を発売している．

アシストがカード型データベースソフトを発売

㈱アシストは，カード型データベース「ゆーぴーカードASKALITE」を発売した．主な特徴は，①表計算ソフトに似た操作性を採用，②あいまい検索機能を装備，③半角・全角にかかわらず，読み方を優先した完全な辞書順の五十音ソートが可能，④1パッケージに5インチと3.5インチの両方のメディアを同梱，⑤シェアウェアのパーソナルカード型データベース「PEMO」を添付――など．同ソフトは，同社の新シリーズ「アシストYUPIシリーズ」の第1弾である．対応機種はPC-9800シリーズおよびPC-286／386シリーズ．価格は9800円．

＊1　家庭内に情報コンセントを設けることで情報通信機器，家電機器，セキュリティ関連機器などを接続できるようにするためのバス規格で，ホームテレホン，エアコン，給湯器などにHBS規格に準拠した製品がある．

アライドテレシスがEthernet用薄型トランシーバを発売

アライドテレシス㈱（☎03-443-5640）は，Ethernet用の薄型8ポートトランシーバ「CentreCOM810」を発売した．主な特徴は，①LEDによるポート状態のモニタリングが可能，②最大8台のCentreCOM810をカスケード接続し，最大64台のワークステーションを接続可能，③幹線のイエローケーブルに接続することも，ミニネットワークを構築することも可能——など．価格は11万9500円．

富士通がスーパーコンピュータを機能強化

富士通㈱は，スーパーコンピュータ「VP2000シリーズ」の機能強化を行ない販売を開始した．今回の強化はUXP／MおよびOS4／MSPのオペレーティングシステムのいずれのオペレーティングシステムでもサポートされる．機能強化した主な点は，①複数プロセッサによる高速・大量処理を実現する4スカラプロセッサ／2ベクタプロセッサの2機種を提供，②1つのプログラムを複数のCPUで分割処理可能な「FORTRAN77 EX／PP」を提供，③制御部にガリウムひ素LSIを採用した32Gbytesのシステム記憶のサポート，④データ転送速度36Mbytes／秒を実現するアレイディスク方式採用の磁気ディスクサブシステムおよび高速光チャンネルの提供，⑤ワークステーションで作成されたC言語プログラムを，VP上でベクタ処理可能にする自動ベクタ化コンパイラの提供，⑥VPとワークステーション間の高速データ転送が可能な1Gbit／秒のマルチベンダーLANをサポート，7 UNIXシステムとの連携強化のためMSPでのTCP／IPサポートの強化——など．価格は7900万円より．

富士通がHBS用8bitマイクロコントローラを発売

富士通㈱は，ホームバスシステム[*1]規格（ET-2101）準拠のホームバス・プロトコルコントローラ「MB89780シリーズ」の2種を富士通VLSI㈱と共同で開発，発売した．今回発売されたのは，EPROM16Kbytesを内蔵した「MB89W785」とワンタイムPROM16Kbytesを内蔵した「MB89P785」．主な特徴は，①F2MC-8のCPUコアを内蔵，②HBPCを内蔵し，プロトコルのハード化による高速処理を実現，③ノイズガード機能を内蔵し，信頼性を向上——など．MB89W785はEPROMを内蔵しており，実機での最終的な機能評価をデバイスレベルで行なうことが可能．MB89P785はワンタイムPROMを内蔵しているため，量産試作や量産立ち上げ時に使用できる．サンプル価格は，MB89W785が1万2000円，MB89P785が3600円．量産時期は1991年12月の予定．

リード・レックスがF-1 DATABOXをバージョンアップ

㈱リード・レックスは，J-3100シリーズ用のカード型データベースソフト「F-1 DATABOX」をバージョンアップし，「同Ver.1.1」として発売した．同ソフトは，従来J-3100対応版とDynaBook版の2製品に分かれていたのを，J-3100対応版として統一したもの．バージョンアップによる主な改良点は，①設計機能の向上，②可変長データと日付・時刻データのサポート，③入力支援機能の充実，④電子手帳とのデータ入出力機能をサポート——など．Ver.1.0で作成したデータは，そのままVer.1.1のデータとして使用できる．価格は3万2000円．

理経が16bitサンプリングA／Dコンバータを発売

㈱理経は，米ANALOGIC社の開発した16bitサンプリングA／Dコンバータ「ADC4300」の販売を開始した．主な特徴は，①16bit分解能と，サンプリングレート50KHzを実現，②±1/2 LSBの微分直線性を実現，③サンプルホールド機能内蔵により外部サンプルホールド回路が不要，④リファレンス回路，クロック回路，出力バッファ回路とセルフキャリブレーション機能を内蔵，⑤アナログ入力範囲は，ユニポーラ0～5V，0～10V，0～20Vと，バイポーラ±2.5V，±5V，±10Vより選択が可能——など．価格は5万3400円より．

＊2　High level Data Link Control. SDLCを基に機能追加を行なったISOで標準化されたデータリンク制御手順のこと.

＊3　Syncronous Data Link Control. IBM社が開発した同期式データリンク制御手順．コンピュータ間で通信を行なう際，通信を開始しようとする局が開局メッセージを送信し，相手の局がこれに応えて肯定応答メッセージを送信するなどの，同期式通信での通信制御手順を規定したもの．

エプソンがPC-9800互換のパソコン2機種を発売

セイコーエプソン㈱は，パーソナルコンピュータ「PC-286C」および「PC-286LP」の2機種を発売した．PC-286Cは本体・キーボード一体型の16bitパーソナルコンピュータで，主な特徴は，①CPUにクロック周波数10MHzの80286を採用，②FM音源3和音，SSG音源3和音，スピーカ内蔵のサウンド機能を装備，③拡張スロットを1個装備し，PC-9800シリーズのオプションボードが使用可能――など．PC-286LPはプリンタ内蔵型パーソナルコンピュータで，主な特徴は，①CPUにクロック周波数10MHzの80286を採用，②3.5インチフロッピーディスクを2台装備，③ディスプレイは8階調640×400ドットのFTN型サイドライト付き白液晶ディスプレイを採用，④80桁48ドットヘッド熱転写プリンタを内蔵，⑤オプションのフォントカードにより毛書体，教科書体，ゴシック体などのフォントを使用可能，⑥PC-286Lシリーズのオプションボードが使用可能な専用拡張スロットを1個装備――など．価格はPC-286Cが16万8000円，PC-286LPが29万8000円．

昭和情報機器がINSネット64を利用したMT転送装置を発売

昭和情報機器㈱（☎03-262-3061）は，INSネット64を利用して遠隔地にMTデータを転送する「ISMT」を発売した．主な特徴は，①64Kbytes／秒で転送が可能，②HDLC手順により確実なデータ転送が可能，③各種のテープフォーマット（JIS，IBMなど）に対応，③1600／800BPI，2400フィートのテープが使用可能――など．価格は200万円．

富士通がデータリンクコントローラ用LSIを発売

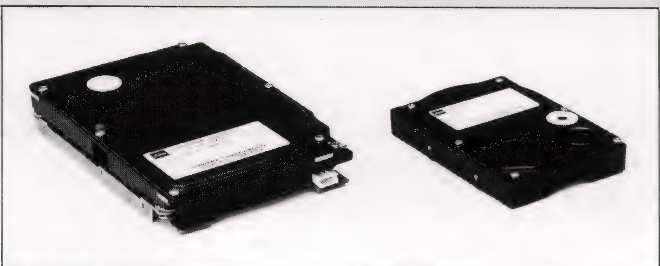
富士通㈱は，富士通デバイス㈱および富士通VLSI㈱と共同でHDLC*2およびSDLC*3をサポートしたデジタル通信機器の通信制御用LSI「MB89374」を開発し，販売を開始した．主な特徴は，①最大2.5Mbytesの通信速度を実現，②受信アドレスフィールドデータ比較機能や，インターフレームフラグ送出機能，任意フラグ送出機能，NRZI符合での自動復帰機能などの通信プロトコル制御機能を内蔵し，ソフトウェアの負担を軽減，③DMAインターフェイス，16bit送信バイトカウンタを内蔵し，外付けDMAコントローラとの接続が容易，④最大75mWと同社従来比3分の1の低消費電力を実現――など．価格は800円．

日商エレクトロニクスがビデオプロジェクタを発売

日商エレクトロニクス㈱（☎03-544-8207）は，カナダのElectro Home社の開発した自動色ずれ調整機能搭載のコンピュータ対応ビデオプロジェクタ「ECP-3100ACON」および「ECP-4100ACON」の2機種の販売を開始した．自動色ずれ調整機能は，テストパターン投影時にプロジェクタ全面下部にあるセンサーカメラによってグリーン・レッド，グリーン・ブルーの各色ずれを検知し，プロジェクタにフィードバックすることにより補正を行なうもの．これにより，これまで手動で行なっていた自動色ずれ調整を自動的に行なうことが可能になった．接続する端末の水平走査周波数は，ECP-3100ACONが15～55KHz，ECP-4100ACONが15～80KHzとなっており，ECP-4100ACONはVTRからパーソナルコンピュータ，ワークステーションなどに接続可能．価格は520万円から．

富士通がベクトルフォントジェネレータを発売

富士通㈱は，ドットデータをアウトラインベクトルデータに変換する機能とアウトラインベクトルデータをドットデータに変換する機能をワンチップで実現したベクトルフォントジェネレータ「MB89511」を発売した．主な特徴は，①イラストなどのドットデータをベクトルデータに変換可能，②ベクトル化されたデータを縦横各2～32767ドットの範囲でドットデータに復元可能，③斜体・回転・鏡像・影付きなどのドットデータが生成可能――など．サンプル価格は2300円．



東芝が国内最大容量の2.5インチHDDなどを発売

㈱東芝は，2.5インチサイズとしては国内最大の43Mbytesの記憶容量を実現したハードディスクドライブ「MK-1122FC」を開発し，同サイズで記憶容量22Mbytesの「MK-1022FC」とともに発売した．また，同時に3.5インチサイズで1インチの薄型化を実現した記憶容量106Mbytesの「MK-1034FC」，同53Mbytesの「MK-1032FC」を発売した．MK-1122FCは，ギャップ長を狭めるために薄膜を用いてヘッドを形成し，高密度に記録された磁気信号を安定状態に保つために磁性体に抗磁力の大きな材料を採用したことなどによって，従来と比較して約3.4倍の記録密度を実現した．MK-1122FCとMK-1022FCは，平均アクセス速度23msを実現し，32Kbytesのキャッシュメモリを内蔵している．MK-1034FCとMK-1032FCは，平均アクセス速度16msを実現し，64Kbytesのキャッシュメモリを内蔵．価格は，MK-1122FCが10万円，MK-1022FCが7万円，MK-1034FCが13万円，MK-1034FCが7万円．

データコントロールがMac用のNFSを発売

データコントロールリミテッド（☎06-763-2662）は，米Wollongong社の開発した，TCP／IPによるファイル／プリンタ共有を行なうネットワークソフトPathWayシリーズのMacintosh対応クライアント「PathWay Client NFS for Macintosh」の発売を開始した．主な特徴は，①NFSアイコンをフォルダーにドラッグするだけでインストール可能，②VAX，Sun，その他のNFS対応システムはディスクドライブアイコンとして使用可能，③常駐メモリサイズは35Kbytesで，NFSサーバのクライアントとしての機能を実現，④マウントされたリモートNFSディレクトリはMacintoshのファイルアイコンとフォルダーアイコンとして使用可能，⑤プリントサーバ・アプリケーションを添付し，他のコンピュータからASCIIテキストおよびPostScriptファイルをApple LaserWriterやHP LaserJetなどのLocalTalkベースのプリンタにプリント可能──など．動作環境は，Macintosh Plus，SE，Mac II，およびIIcx，メモリ1Mbytes以上，MacOS System6.0以上が必要．価格は5万4000円．

富士ゼロックスが写楽とパソコンを接続するアダプタを発売

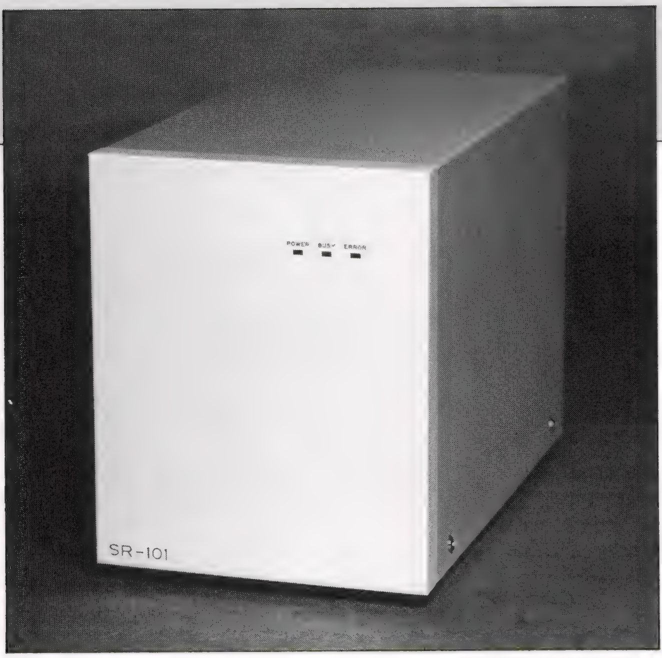
富士ゼロックス㈱は，同社のハンディ転写マシン・写楽IIIとJ-3100シリーズなどを接続し，写楽IIIで読み取った画像のパーソナルコンピュータの上で編集・加工，またフロッピーへの保存を可能にする専用アダプタ「写楽PA 2」を発売した．対応機種は，J-3100およびJ-3300シリーズ．動作環境は，MS-DOS Ver.3.1以上，380 Kbytes以上の空きメモリが必要．価格は2万9800円．

日商エレクトロニクスがNEWS対応HDDを発売

日商エレクトロニクス㈱（☎03-544-8441）は，ソニーNEWSシリーズ対応の640MbytesSCSI拡張ハードディスクユニット「NE-640」を発売した．同ユニットは，平均アクセス時間16msを実現しており，ソニー製のハードディスクユニットにフルコンパチブルなため専用のデバイスドライバは不要．動作環境はNEWS-OS3.3以上が必要．価格は99万8000円．

Apple社がAdobe社と新たなライセンス契約を締結

米Adobe Systems社は，米Apple Computer社と新規の技術ライセンス契約に関するレターオフインテントの締結を発表した．この合意内容は，今後両社は互いの技術（Adobe社のPostScript技術とApple社のプリンタ技術）を盛り込んだ新しい製品群を協力して開発していくということであるが，それ以上のアナウンスはされていない．両社の関係については，Apple社はAdobe社のDisplayPostScriptを採用しないと発表し，さらに続いて，Apple社がAdobe社の株を全部放出して以来冷えきっていると見られていたので，今回の発表は業界にざわめきを与えている．特に今年3月に始まったフォント戦争はこの状態に拍車をかけているように見えたが，消息筋の間では，Microsoft社のTrue Typeフォントは難航しており，サポートするソフト開発者に大きな負担がかかる，ともいわれている．実際にApple社がTrue Typeを含めて出荷するとアナウンスしているSystem7.0の出荷は遅れに遅れている．その原因の1つがTrue Typeにあるのではないか，との憶測もされているのは事実で，このあたりに変わり身の早いApple社の計算が働いているのかもしれない．真相のほどは不明にしても，Adobe社のType 1フォーマットの地位はますますゆるぎないものになると思われる．



リンクスがWS用のRAMディスク装置を発売

㈱リンクスコーポレーションは，SCSIインターフェイスを採用したRAMディスク「RD-900」シリーズを発売した．RD-900シリーズは平均アクセス時間0.3ms，転送速度3.0Mbytes／秒を実現し，RD-900シリーズを利用することにより，通常のハードディスク装置と比較してシーケンシャルファイルアクセスで3～5倍，ランダムアクセスで10～15倍程度のディスク入出力速度の改善が可能．対応機種は，Sunシリーズ，VAXシリーズ，DECstationシリーズ，HP9000シリーズ，AVシリーズ，NEWSシリーズ．価格は，ディスク容量が96Mbytesのものが280万円，同128Mbytesのものが360万円，同256Mbytesのものが700万円．

インテルジャパンなど7社がDVIコンソーシアムを設立

インテルジャパン㈱は，DVIの普及を目指し7社の発起人とともにDVIコンソーシアム「DAVIS (Digital Audio Visual Interactivemedia Society)」を発足し，第1回総会を開催した．DAVISでは，今後DVIの仕様の国際標準化や応用ハードの互換性を促進していく．会長には，慶應大学の渡辺彰計測工学部教授を据え，マルチメディア研究と情報交換の場を提供し，DVI技術を国際的なレベルで確立しようというもの．具体的にはセミナーや研究会の開催，情報誌発行などの活動が主となる．設立発起人は，伊藤忠テクノサイエンス，沖電気，大日本印刷，電通プロックス，日本アイ・ビー・エム，ハドソン，インテルジャパンの7社，発足時の参加社は約50社．

ヤマハがフロッピーディスクファクシミリを発売

ヤマハ㈱は，フロッピーディスクドライブと電話機を一体化した通信端末装置を送受信側に設置し，ISDNを利用してフロッピーディスクのデータの送受信を行なうフロッピーディスクファクシミリ「FDX 1」を発売した．主な特徴は，①フロッピーディスクのデータを最大約3分で転送可能，②3.5インチおよび5インチの2ドライブを搭載し，市販の8インチドライブも接続可能，③ディスクのフォーマットはJIS規格（IBM／ISOフォーマット）に準拠しているものに対応，④ISDNを利用することにより，通信回線上のエラー発生率を約10万分の1に抑えることが可能，⑤データ転送中に電話をかけることも可能，⑥相手先のデータをこちらから要求して受信できるポーリング機能を装備，⑦一度受信したデータに2度書き込みを防止する機能を装備，⑧パスワード・ID認識などのセキュリティ機能を装備，⑨1度の登録で複数個所へデータを送る順次同報機能を装備――など．価格は49万8000円．

オムロンがUNIX用のファイリングシステムを発売

オムロン㈱は，㈱エイコットの開発した，文書・図・表などのイメージ情報を管理するファイリングシステム「HOTDOC」をLUNAに移植して発売した．主な特徴は，①X Window上で動作し，マウスとアイコンによる検索や編集などの操作が可能，②文書は，キャビネット・フォルダ・文書の階層構造で管理可能，③フリーハンド曲線・円・イメージなどの切り張りや反転表示などの文書イメージ編集機能を装備，④NFSに対応し，グループでの共同利用やリモート検索か可能――など．価格は55万円．

オーキッドがSun用のスクリーンエディタを発売

㈱オーキッド（☎03-827-7496）は，Sun用の高速スクリーンエディタ「MAX EDITOR」を発売した．主な特徴は，①直接画面表示によりスクロールなどの高速画面表示が可能，②マルチファイル，デュアルウィンドウを採用，③Sunのタイプ4キーボードの特殊キーにコマンドを割り当てることが可能，④コンパイラのエラーメッセージによるタグジャンプ機能を装備，⑤カーネルからのメッセージをエディタに取り込むことが可能，⑥UNIXの正規表現での文字列検索が可能，⑦異常終了時の編集テキストセーブ機能を装備――など．価格は9万7000円．

アイワが電池で使用可能なモデムを発売

アイワ㈱は，電池で使用可能な2400bpsモデム「PV-M24VM 5」と1200bpsモデム「PV-M12」の2機種を発売した．主な特徴は，①アルカリ電池使用時にM24VM 5では約8時間，M12では約20時間の通信が可能，②パワースイッチをオートにすることにより，通信時のみモデム電源が自動的にONになり，無駄な電池の消耗を防ぐことが可能，③AA／MA／MMの3モードのNCUを装備，④トーン／パルスを自動的に切り替える機能を装備，⑤M24VM 5は，通信エラー制御としてCCITT V.42とMNPクラス5を搭載，⑥内蔵バッファとフロー制御でターミナル速度固定での通信が可能，⑦モデムや通信の設定状況を表示するモニタ機能を装備，⑧リモートメンテナンス機能を搭載――など．価格は，PV-M24VM 5が4万2800円，PV-M12が1万9800円．

LIFEBOATがVAX用RAMディスクを発売

㈱LIFEBOAT (☎03-293-4717) は，米EEC Systems社の開発した，VAXの主記憶の一部を通常のディスクと同じように使用できる仮想ディスク生成システムソフトウェア「TURBODISK」を発売した．主な特徴は，①実メモリ上でのRAMディスクのエミュレートおよび実際のディスク上の連続領域を使用したディスクのエミュレートが可能，②ディスクのファイルアクセス情報の集計が可能，③同期を取りながら同じ内容をディスクに書き込む機能を装備，④ユーザーが決めたインターバルでメモリディスクの内容をハードディスクにバックアップする保護機能を装備――など．価格は120万円より．

富士通が1990年代のメインフレームコンセプトを発表

富士通㈱は，1990年代のメインフレームシステムコンセプト「MISSION／DC」を発表した．MISSION／DCは，'90年代に期待されるメインフレームの役割を明確にし，この役割に対応する機能群を論理的な，データベースドメイン(DP)，アプリケーションドメイン (AP)，コミュニケーションドメイン (CP) の3つのドメインと，それを統合管理するマネージメントドメイン (MP) に分け，これらのドメインを物理的な単一システムおよび複数システム上で組み合わせ，トータルとしてメインフレームを実現できるようにするもの．同社はこのコンセプトを実現する製品の第一弾として，①世界最高速の処理性能を実現した大型汎用コンピュータ「M-1800モデルグループ」，②高速バッチ処理を実現する「エクセルバッチ」，③大量のデータ管理・ストレージ装置の有効利用のための「統合ストレージ管理機能GSM／MNGR」，④複数のシステムの集中監視・制御およびリモート運用を実現する統合管理システム「FTOPS」――などの製品を提供する．M-1800モデルグループの価格は1カ月7800万円より．

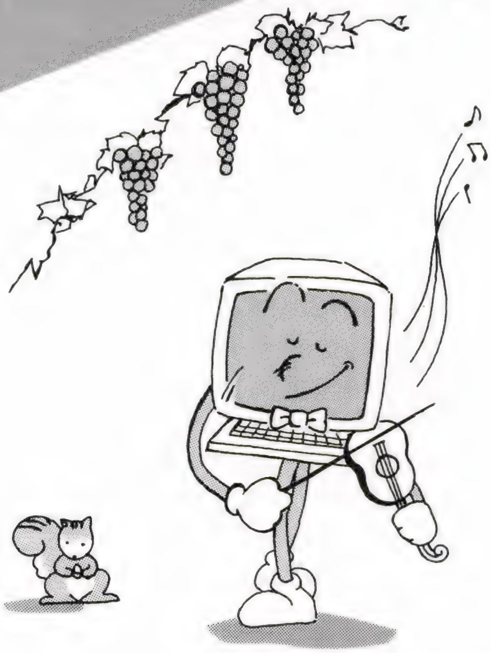
理経がシミュレータ用のビジュアルシステムを発売

㈱理経は，米E&S社の開発したコンピュータ模擬視界発生装置「ESIG-3000」および「ESIG-4000」の2機種を発売した．ESIG-4000の主な特徴は，①フルカラー写真テクスチャーによる実物との相関のとれた地形や構造物の表示が可能，②レンジバッファ方式による正確で汎用性の高い視界プライオリティの決定が可能，③遠景から近景へのなめらかな映像ディティールの変更を実現するビトウィーニング技法を採用――など．ESIG-3000は，ESIG-4000に採用されたレンジバッファ方式と，フルカラーグローバルテキスチャー機能を装備しており，モジュラ構成を採用した．価格はESIG-3000が4億円程度から．

エプソンがAdobeとPostScriptのライセンス契約を締結

セイコーエプソン㈱は，米Adobe Systems社とPostScriptソフトウェアのライセンス契約を締結したと発表した．エプソンによれば，1991年第1四半期には海外市場向けにIBM PCやMacintosh用のPostScriptレーザープリンタを発売する予定とのこと．

イベントカレンダー



10 10·17 ― 10·31

1990 OCT		
17	水	UNIX MAGAZINE セミナー（大東京火災新宿ホール・東京） The Online Audiovisual Catalogers Conference（N.Y., U.S.A.）
18	木	第36回KOBEオフィスフェア（サンボーホール・神戸） '90 文字と画像の未来展（科学技術館・東京）
19	金	
20	土	
21	日	INTELEC '90（Orlando, Fla.,U.S.A.）
22	月	データショウ'90（国際見本市会場・東京） 31st Annual Foundations of Computer Science（U.S.A.）
23	火	セミコンジャパン '90（幕張メッセ・千葉） INTER COMM '90（Vancouver, B.C.,Canada）
24	水	第6回ヒューマン・インターフェース・シンポジウム（東京） CASExpo（Hong Kong Convention & Exhibition Centre）
25	木	第27回情報科学技術研究集会（大阪国際交流センター・大阪） NORTHEAST COMPUTER SHOW（Boston, Mass.,U.S.A.）
26	金	
27	土	
28	日	NACLP '90（Austin, Tex.,U.S.A.）
29	月	
30	火	第4回バーコードおよびIDシステム展（流通センター・東京） '90中部エレクトロニクスショー（中小企業振興会館・名古屋）
31	水	UNICOM'90（Washington, D.C.,U.S.A.） UNIX Expo 1990（New York, N.Y.,U.S.A.）

国内イベント

10月

●は開催地◆は問い合わせ先を示す

17, 18

UNIX MAGAZINE セミナー

●大東京火災新宿ホール・東京

◆(株)アスキー直販部

〒107-24 港区南青山6-11-1

18-20

'90 文字と画像の未来展

●科学技術館・東京

◆印刷之世界社 (03)5256-0591

22-25

データショウ '90

●国際見本市会場・東京

◆(社)日本電子工業振興協会

(03)433-4547

24-26

第6回ヒューマン・インターフェース・シンポジウム

●総評会館・東京

◆ヒューマン・インターフェース・シンポジウム事務局

(0468)59-2328

11月

1, 2

マイコンツールフェア '90

●池袋サンシャインシティ・東京

◆日本システムハウス協会事務局

(03)668-3151

6-9

NICOGRAPH'90

●池袋サンシャインシティ・東京

◆日本コンピュータ・グラフィックス協会 (03)233-3475

6-9

ISS '90 第三回国際超電導シンポジウム

●ホテルメトロポリタン・仙台

◆国際超電導産業技術研究センター

14-16

'90西日本エレクトロニクスショー／九州アジアテクノフェア'90

●西日本総合展示場・北九州

◆九州産業技術センター

(092)411-7391

15, 16

第16回jusUNIXシンポジウムおよび展示会

●大阪リバーサイドホテル・大阪

◆日本UNIXユーザー会事務局

(03)356-0156

18-22

ICSLP 90:

1990年音声言語処理国際会議

●神戸国際会議場・神戸

◆(株)サイマルインタナショナル内

ICSLP 90事務局 03(586)8691

27, 28

'90文書情報処理システム展

●東京都立産業貿易センター・東京

◆日本経営協会 (03)403-1331

28-30

DECWORLD'90 Japan

●東京ベイNKホールほか・東京

◆日本DEC企画本部広報部

(03)228-4480

12月

4, 5

UNIX Fair'90

●新宿NSビル・東京

◆日本UNIXユーザー会

(03)237-1789

11 1990 NOV

11・1 ▶ 11・30

日	曜	イベント	日	曜	イベント
1	木	マイコンツールフェア '90（池袋サンシャインシティ・東京） Fifth Annual Workshop on computer Communications（U.S.A.）	17	土	
2	金		18	日	ICSLP'90: 1990年音声言語処理国際会議（神戸国際会議場・神戸）
3	土	VIST 90: 3rd Symposium on User Interface Software （Snowbird, Ut.,U.S.A.）	19	月	
4	日	Fourth IEEE Workshop on Metropolitan Area Networks（U.S.A.）	20	火	
5	月	NICOGRAPH '90（有楽町マリオン・東京） 国際エレクトロニック・シネマ・フェスティバル（幕張・千葉）	21	水	グラフィックスとCADシンポジウム（機械振興会館・東京）
6	火	Software Development '90 Fall（Boston, Mass.,U.S.A.） NICOGRAPH '90（池袋サンシャインシティ・東京）	22	木	
7	水	第二回オプトエレクトロニクス国際会議・見本市（神戸） Internepcon Singapore'90 Exhibition&Conference（Singapore）	23	金	
8	木	International Workshop on Network and Operating System Support for Digital Audio and Video（Berkeley, U.S.A.）	24	土	
9	金	ハイテク広島 '90（広島県立広島産業会館・広島）	25	日	
10	土		26	月	CSM '90: Conference on Software Maintenance （San Diego, Calif.,U.S.A.）
11	日		27	火	'90 文書情報処理システム展（東京都立産業貿易センター・東京） MICRO 23（Orland, Fla.,U.S.A.）
12	月	Supercomputing '90（New York, N.Y.,U.S.A.） COMDEX/Fall '90（Las Vegas, Nev.,U.S.A.）	28	水	DECWORLD '90 Japan（東京ベイNKホールほか・東京） TOOL'90: 第1回国際ソフト&データベース管理・会議（Germany）
13	火	Wescon/90（Anaheim, Calif.,U.S.A.）	29	木	
14	水	PRICAI '90: 環太平洋人工知能国際会議（名古屋国際会議場） '90西日本エレクトロニクスショー（西日本総合展示場・北九州）	30	金	
15	木	第16回jusUNIXシンポジウムおよび展示会 （大阪リバーサイドホテル・大阪）			
16	金				

海外イベント

10月

●は開催地◆は問い合わせ先を示す

21-24

INTELEC '90

●Orlando, Fla., U.S.A.

◆Jim Fletcher, 41 Robbins Ave., Berkeley Heights, NJ, 07922 (201)464-7919

22-26

国際コンピュータ統合化専門見本市・会議

●Munchen, Germany Munchen Messegelande

◆在日ドイツ商工会議所見本市部 (03)593-1641

22-26

EUUG Autumn '90 Conference and Exhibition

●Nice, France.

◆Hele Gibbons, The Secretariat, EUUG, Owles Hall, Buntingford, Hertfordshire SG9 9PL. 44 763-73039

23-26

INTER COMM'90

●Vancouver, B.C., Canada

◆INTER COMM Office P.O. Box 1616, Blaine, WA 98230

25-27

NORTHEAST COMPUTER SHOW

●Boston, Mass., U.S.A. Bayside Expo Center

◆ザ・インターフェイス・グループ・インク日本支社 (03)237-8759

31-11/2

UNIX Expo 1990

●New York, N.Y., U.S.A.

◆National Expositions Co. Inc., 212-391-9111

11月

6-8

SOFTA 90: Trade Fair on Computer Software

●Helsinki, Finland Old Harbour

◆Helsinki Fair Ltd 3580 173-341

7-10

Internepcon Singapore'90 Exhibition & Conference

●Singapore Changi International Exhibition Center

◆カーナーズ・エクスポジション・ジャパン (03)349-8501

8-13

EXPO COMM CHINA: 中国国際電気通信情報展・会議

●北京, 中国 China International Exhibition Center

◆E.J.クラウスアンドアソシエート社 (03)584-1548

12-16

COMDEX/Fall'90

●Las Vegas, Nev., U.S.A. Las Vegas Convention Center

◆ザ・インターフェース・グループ・インク日本支社 (03)237-8759

12月

2-5

Computer Graphics For Design/Fall

12 1990 DEC

12・1 ▶ 12・31

1	土		17	月	
2	日	GLOBECOM'90 (San Diego, Calif.,U.S.A.) Computer Graphics For Design/Fall (New York, N.Y.,U.S.A.)	18	火	
3	月	Sun User Group Conference & Exhibition (San jose, Calif.,U.S.A.)	19	水	
4	火	UNIX Fair '90 (新宿NSビル・東京) SEARCC '90: SouthEast Asia Regional Computer (Philippine)	20	木	
5	水	セミコン / Korea '90 (Soul, Korea) Communication Networks/Korea (Soul, Korea)	21	金	
6	木		22	土	
7	金		23	日	
8	土		24	月	
9	日	SPDP: Second IEEE Symposium on Parallel and Distributed Processing (Dallas, Tex.,U.S.A.)	25	火	
10	月		26	水	
11	火	'90 国際画像機器展 (東京都立産業貿易会館・東京) Workshop on Heterogeneous Database Systems (U.S.A.)	27	木	
12	水		28	金	
13	木		29	土	
14	金	学生のための総合技術展 (国際見本市会場・東京)	30	日	
15	土		31	月	
16	日				

●New York, N.Y.,U.S.A.
◆Barbara Jeffe (914)741-2850

3-5

Sun User Group Conference & Exhibition

●San jose, Calif.,U.S.A.
◆Sun User Group Office
(415)336-0564

5-8

Communication Networks/Korea

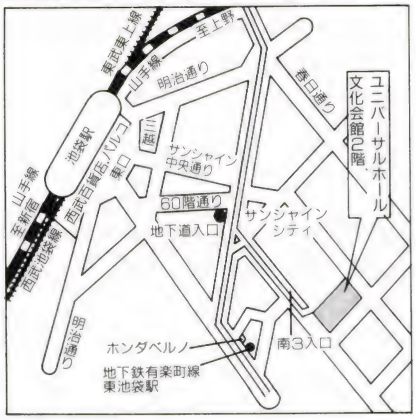
●Soul, Korea.
Korea Exhibition Center
◆IDG ワールドエキスポジャパン
(03)222-6411

9-13

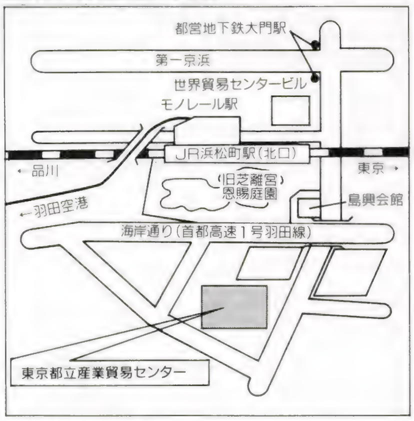
SPDP:
Second IEEE Symposium on Parallel and Distributed Processing

●Dallas, Tex.,U.S.A.
The Colony Parks Hotel
◆Behrooz Shirazi, Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas 75275-0122 ;
214-692-2874

■池袋サンシャインシティ

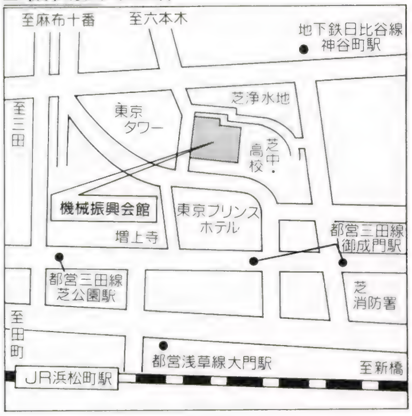


■東京都立産業貿易センター



■総評会館

■機械振興会館



特集

RISCはなぜ速いのか

太田昌孝

CG illustration : MINAKO MIYAJIMA

ワークステーションのCPUパワーは，大型汎用コンピュータ，さらにスーパーコンピュータを脅かすまでに伸びてきた．もはや大型汎用コンピュータをCPUパワーを目的として使う価値はなくなっている．この発展を支えているのがRISC技術である．しかしRISCは，単に命令セットを簡略化しただけのコンピュータではない．確かに命令セットの簡略化には，命令実行時間の短縮という効果はあるが，RISCとはメモリやディスクなどまで含めたコンピュータ全体にかかわる技術なのである．ではなぜCPUパワー，すなわち計算能力を強化するために，コンピュータ全体を考える必要があるのだろうか．また命令セットの簡略化でないとしたら，RISCとは何なのだろうか．コンピュータの成り立ちを原点から考え直すことにより，これらの疑問に答えていく．その過程で，RISCはなぜ速いのか，という最も重要な問題の答えも得られるはずである．

1 RISCとは何か

RISCとは，Reduced Instruciton Set Computerの略であり，CISC(Complex Instruction Set Computer)と対照的な概念である．このRISCが，どのような発想で作られているかを，RISCとCISCの差を対比させつつ，コンピュータの成り立ちとコンピュータのシステム全体としての性能向上の方法として見ていく．

コンピュータとは何か

RISCにしろCISCにしろ，コンピュータは，そもそもどのような機械を実現しようとして作られていたのかについて，コンピュータのデザインの原点から考える．そのうえでフォンノイマンボトルネックによって規定される，ストアードプログラム方式のコンピュータの高速化について検討していく．

コンピュータとは何か

現在のコンピュータは基本的に，フォンノイマンによって発展させられたストアードプログラム方式という原理に基づいて動作する．コンピュータは，大きく分けてCPU (Central Processing Unit，中央処理装置)，メモリ (memory，記憶装置)，I／O (Input／Output，入出力)の3種類の構成要素からなる(図1)．CPUはこれらの要素の制御を行なう．

ストアードプログラムの考え方では，プログラムやデータは，すべてメモリに収納されている．CPUは順次メモリから命令を読み出し，その命令に従ってデータをメモリに書き込んだり，メモリから読み出したりする．つまり，コンピュータの動作は，メモリとCPUとの間のデータ転送によって起きるわけである．CPU自体にはあまりたくさんの情報は保持されないので，計算結果は最終的にはメモリに格納される．したがって，CPUとメモリの結合は重要である．

この結合部分が，CPUの処理能力を実質的に制限してしまうため，フォンノイマンボトルネックと呼ばれている．CPUの8ビットとか32ビットとかいう類別は，フォンノイマンボトルネックのデータ転送のための配線の数のことである．配線が多いほど，そのコンピュータが一時に転送できるデータの量が多くなり，処理も高速になる．

コンピュータの高速化のためには，フォンノイマンボトルネックを拡大する方向と，フォンノイマンボトルネックを通るデータの量を減らす方向の2つのアプローチがある．

CPUとはどういうものか

図2は，実用上最も簡単なCPUのブロック図である．この例ではCPUはアキュムレータ(Accumulator，累算機)，ALU (Arithmetic Logic Unit，算術論理ユニット)，PC(Program Counter)，インス

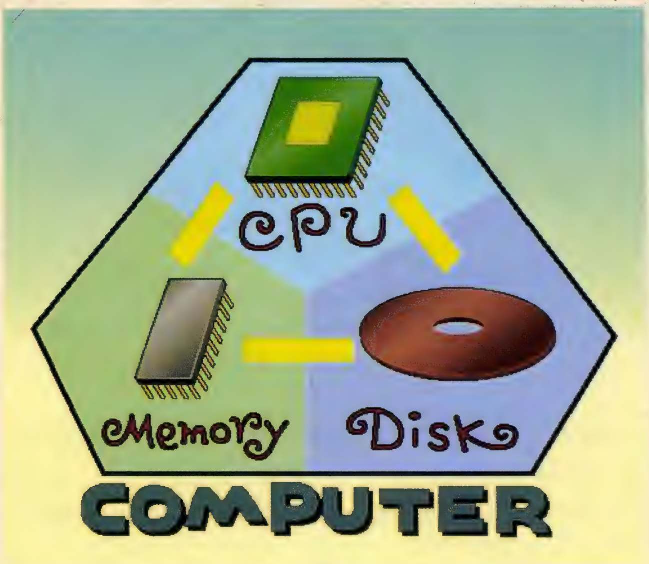


図1　メモリは，コンピュータ内部で情報をbit単位で記憶する要素であり，情報の格納場所は，アドレス(番地)と呼ばれる番号によって指定され，アドレスを与えることにより8bit単位や32bit単位などで検索される．I／Oは，コンピュータ本体と外部との情報のやりとりを行なう．

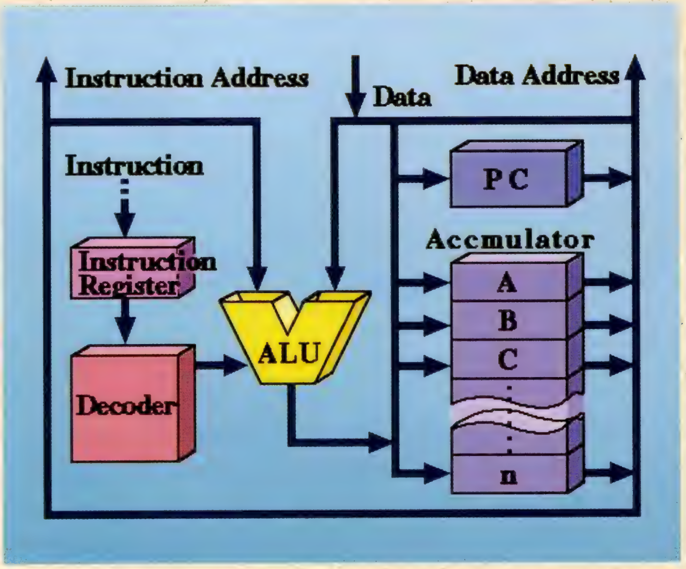


図2　デコーダはインストラクションレジスタを通してPCの指示する命令を読み出し，それに従ってデータをメモリから読み，ALUを制御して計算させ，さらにその結果をメモリに書き出す．PCはその間に次の命令のアドレスを指示している．命令によってはPCを変更することもできるようになっている．

トラクションレジスタ(Instruction Register) およびデコーダ(Decoder)よりなる．

PCは，実行すべき命令のアドレスを保持するレジスタである．通常の場合，命令はアドレスの順番に実行されるので，PCは単なるレジスタではなくカウンタとなっており，命令の実行につれて順次増加していく．リセット時にはPCには0番地などの特定の値がセットされ，そこにある命令から実行が開始される．インストラクションレジスタは，読み込んだ命令を，その実行の間保持しておくものである．ALUは，そのCPUにとっての基本的な演算を行なうためのユニットである．

ALUで演算を行なうためには，データの入力元と，出力先を指定してやる必要がある．そのためには，命令中に直接演算対象のデータのメモリアドレスを指定してもよい．しかし，加算，減算など一般の2項演算では，2つの入力元と1つの出力先の，3つのアドレスを指定する必要があり，命令が長いものになってしまう．この長い命令は，フォンノイマンボトルネックを通じて読まれる．さらにデータ自体の読み出し／書き込みにもフォンノイマンボトルネックが使われる．したがって，実行速度もフォンノイマンボトルネックにより制限される．

そこで，フォンノイマンボトルネックを通るデータを減らすために，演算を毎回メモリ同士でやらずに，片方の入力や，出力にはCPU内部のレジスタを使うことにする．これがアキュムレータである．アキュムレータの利用により，命令では直接には1つのアドレスを指定すればよい．結果をメモリに格納するのは，一般にストア命令と呼ばれる特定の命令を用意しておく．しかし，連続する演算では直前の演算の出力をそのまま次の演算の入力とすることが多い．アキュムレータ(累算機)の名前は，次々と加算を繰り返し数値を累算していくという典型的な利用形態にちなんでつけられているが，そのような場合，途中結果をCPUからメモリにいちいち転送する必要もなくなる．

デコーダでは，命令の内容に応じて，ALUに適切な演算を指定し，アキュムレータやメモリのデータの読み書き，PCの値の更新や次の命令の取り出しなどの制御信号を順番に発生する．

より複雑なCPUでは，より多種類多数の内部レジスタを持ち，メモリとのデータ転送を行なうことなくCPU内部により多くの情報を蓄積できる．

たとえば，データのアドレス指定を簡単に行なうために，インデックスレジスタ（Index Register）が使われる．インデックスレジスタには，メモリのアドレスを入れ，命令にはアドレス指定のフィールドとして何bitかの幅で，そこからのずれ(offset)を設定する．同時に扱うデータは近いアドレスに存在することが多いので，インデックスレジスタにロードされたアドレスを何回も有効に使うことができる．

また，何層にも行なわれるサブルーチンコールの管理を楽に行なうために，スタックポインタ（Stack Pointer）と呼ばれるレジスタが，スタックの現在使用している場所を指すために用いられる．サブルーチンコールを行なう命令では，サブルーチンの先頭アドレスが新たにPCにセットされる前に，古いPCの値がスタックのスタックポインタの指す場所に自動的に格納され，スタックポインタは次の空き番地を指すように更新される．

CPUが複雑になってくると，種々の専用レジスタを持つようになるが，その結果全体の見通しが悪くなる．また，演算の対象となるデータとアドレスのそれぞれに専用のレジスタを使うと，ALUによる乗算や加算の結果をアドレスとして使うような場合に，アキュムレータからインデックスレジスタにわざわざデータを転送する必要がおきる．そこで，数値の格納やアドレスの指定の補助の機能を併せ持つレジスタを用意し，ゼネラルレジスタ（汎用レジスタ）と呼ぶ．もちろん本来複数あったレジスタの代用には1つだけでは不足するので，ゼネラルレジスタは複数（普通4～32個程度）存在し，まとめてレジスタファイルと呼ばれる．レジスタファイル中のどのレジスタがどのように使用されるかは，そのつど命令で指定する．

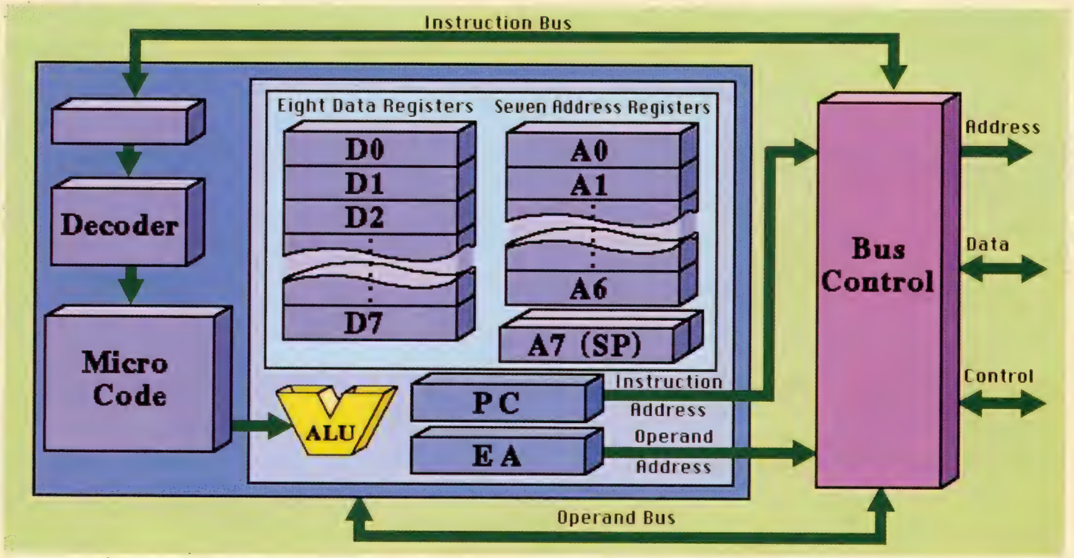


図3　典型的なCISCである68000のブロック図．68000は16個のゼネラルレジスタを持つが，そのうち8個はもっぱら数値，8個はもっぱらアドレスの値を扱うのが便利なようになっている．アドレスレジスタの1つは，スタックポインタとしても使われる．

命令セットの発展

一般にRISCとCISCとを比較するとき，命令セットが最も異なる点とされる．まずはCISCの思想と現状を探るべく，原始的な命令セットを持つコンピュータから強力な命令セットを持つCISCへの発展，さらに命令セットが発展した原因，またこの発展の理由づけについて考えていく．

命令のフォーマット

普通の言語に，主語や目的語のような動作の対象と動詞のような動作の内容が存在するように，コンピュータの命令も，その働きを示すオペコード（Operation Code）と呼ばれる部分と，動作の対象を示すオペランド（Operand）と呼ばれる部分からなる．オペコードは，もっぱら，ALUのようにデータを加工する部分の動作の指定に関係し，オペランドは，レジスタ，メモリのようにデータを格納する部分の指定に関係する．1つの命令中にはオペコードは1つだが，オペランドは，命令の内容に応じて複数存在する．

ここで最も単純な命令体系を持つコンピュータとして，減算命令を1つ持つだけのコンピュータを考えてみよう．命令としては，

```
SUB  A,B,C
```

だけを持ち，メモリのアドレスAからアドレスBの内容を引き，結果が0でなければアドレスCにジャンプ，0なら次のアドレスの命令を実行する．Cを省略した場合は，次のアドレスを指定したものとする．

このような単純なコンピュータでも，アドレスAの内容をアドレスBに移すことができる．アドレスCを作業領域に利用して，

```
SUB  B,B
SUB  C,C
SUB  C,A
SUB  B,C
```

とすればよいし，アドレスAとアドレスBの内容の加算を行ない，結果をアドレスAに入れるには，アドレスCを作業領域に利用して，

```
SUB  C,C
SUB  C,B
SUB  A,C
```

とすればよい．乗算なら，アドレスBが0でなく，アドレスCを作業領域に利用して，アドレス0には1が入っているものとすると，

```
     SUB  C,C
LOOP SUB  C,A
     SUB  B,0,LOOP
     SUB  A,A
     SUB  A,C
```

で可能である．このようにして，命令が1つだけのコンピュータでも，その組み合わせでいかなる複雑な操作も記述することはできる．

CISCの発展

1命令コンピュータにも長所はある．それは，オペコードがいらないことである．なにせすべての命令が同じなので，いちいち命令の種類を指定する必要はないのだ．ところが，オペランドのほうはというと，毎回データ2つとジャンプ先1つを指定しなければいけないので，あまりに無駄が多い．また，実行速度の点でも，減算だけでなく基本的な演算がいろいろそろっていたほうが，当然効率がよい．

プログラムの処理速度を上げる方法の1つは，必要な処理に対して専用の命令を用意することである．さらに，それぞれの命令に対して専用のハードウェアを用意すれば，今まで組み合わせで実現していた処理を，より多くのハードウェアを並列に動作させることにより即座に実行できるようになる．そこで，いろいろな専用ハードウェアを用意し，それに対応する命令を増やしていくと，全体の能率が向上することが期待できる．

まず，演算命令としては，加減乗除やシフトなど，ALUでできる演算に対応した命令を取りそろえることになる．

データ移動の命令としては，各種レジスタ間，レジスタとメモリの間は最低限必要であるし，メモリとメモリの間の直接転送命令も場合によっては望ましいかもしれない．

制御に関連する命令としては，PCになにかの値をロードする無条件ジャンプ命令が必要である．また，演算の結果が0とか負だとかの条件により，PCをそのままインクリメントするか他の値をロードするかを変化させる，条件ジャンプ命令も必要である．また，ジャンプだけでなくサブルーチンを呼び出すためのスタック操作を行なってくれるコール命令も，必需品である．

また，よく組み合わせて使われる命令は，それらを組み合わせて1つの命令に

表1●標準的な命令構成

演算	add	r4, r3	レジスタ3にレジスタ4の値を加算する．符号の有無，ワードの長さ，桁上がりなどの条件によって命令が変わる．
	lsl	#3, r3	レジスタ3を左に3ビットシフトする．あふれたビットの扱いによって論理シフト，算術シフトなどに分類される．
データ移動	move	r7, −(sp)	レジスタ7からスタックポインタの指しているアドレスにデータを移動し，デクリメントする．スタックのプッシュ．
制御	jmp	label	labelというラベルへの無条件ジャンプ．アドレスを指定できる範囲に制限があるのが普通である．
	jsr	printf	サブルーチンへのジャンプ．プログラムカウンタをスタックに退避し，制御を指定したアドレスに移す．
比較	cmpa	#7777, r5	16進7777と，レジスタ5との比較．比較した結果はフラグに反映し，次に続く条件ブランチの行き先に影響を及ぼす．
複合命令	loop	label	専用のレジスタをデクリメントし，その結果に応じてラベルにジャンプするかどうか決める．
複雑な命令	pusha		すべてのレジスタをまとめてスタックに退避する．popaという命令によってまとめて復元できる．

68000や80386など，パソコンレベルで使われているプロセッサでも，この程度の命令は用意されている．実際にはこれらの命令にアクセスするデータの長さ，アドレスの指定の仕方などの指定が加わって，もう少し複雑になる．

してしまう．たとえば，レジスタに最初，ループの回数を設定して，ループを実行するたびに最後にレジスタの内容を1減らし，それが0でない限りループの頭に戻るというのは，よくある操作である．そこで，「レジスタの内容を1減らしてそれが0でない場合条件ジャンプする」という複合命令は，よく現われる．

極めて複雑な複合命令の例としては，複数のレジスタをまとめてメモリに転送する命令や，N次の多項式の値を計算する命令などもある．

可変長命令

専用のハードウェアと命令をあまりに増やし過ぎると，命令の長さが短いコンピュータでは，命令のオペコードが長くなりすぎる．そこで，可変長の命令という考えが生じる．

8ビットのコンピュータで，アドレスの指定に16ビット必要だとすると，減算のみのコンピュータでは，1つの命令を読み込むだけで6回のメモリアクセスが必要となる．また，演算の種類が256種類あっても，オペランドが常に16ビットのアドレス1つだと，それだけでも3回のメモリアクセスが必要である．一方命令によっては，レジスタ間のデータ転送のように頻繁に使われるがオペランドはレジスタを指定するだけで，オペコードとあわせて1バイトあれば十分な場合もある．そこで，よく使われるオペコードやオペランドには短い表現を与え，あまり使われないオペコードやオペランドはある程度長くするというのが，可変長の命令の考え方である．可変長の命令の採用により，命令の読み込みの量は減少する．

オペランドは，アドレスモードの考え方を持ち込めば可変長にできる．たとえば加算を行ないたい場合，その入力や出力はレジスタであったり，メモリであったりする．レジスタ同士の加算しかない場合にメモリ内のデータを加算するには，ロード命令により一度レジスタにデータを読み込んでから加算命令を発行することになるが，このメモリアクセスを加算命令に複合することもできる．これがアドレスモードである．

典型的なアドレスモードの指定方法は，命令中でオペランドを指定する数ビットの部分を2つに分け，片方をアドレスモード，片方をレジスタに使う．アドレスモードには次のようなものがある．

レジスタ　レジスタの内容がそのままオペランドになる．

レジスタ間接　レジスタの内容のアドレスがオペランドになる．

イメディエイト　命令に続く語がそのままオペランドになる．

間接　命令に続く語がオペランドのアドレスになる．

インデックス　レジスタの内容に，命令に続く語を足したアドレスがオペランドになる．

ポストインクリメント　レジスタの内容のアドレスがオペランドになる．レジスタの内容は次のアドレスを指すように増加される．

プリデクリメント　レジスタの内容が，直前のアドレスを指すように減少させられ，減少後のアドレスがオペランドになる．

ポストインクリメントやプリデクリメントはCの＋＋や－－の演算子に対応しており，ブロック転送などを行なう場合に便利である．

CISCでは，一時，命令の直交性という考えがもてはやされたが，これは，すべてのオペコードがすべてのアドレスモードと組み合わせて利用できるということである．直交性は，アセンブラ言語を覚えるときに，細かな例外を記憶しなくてすむのがありがたい．

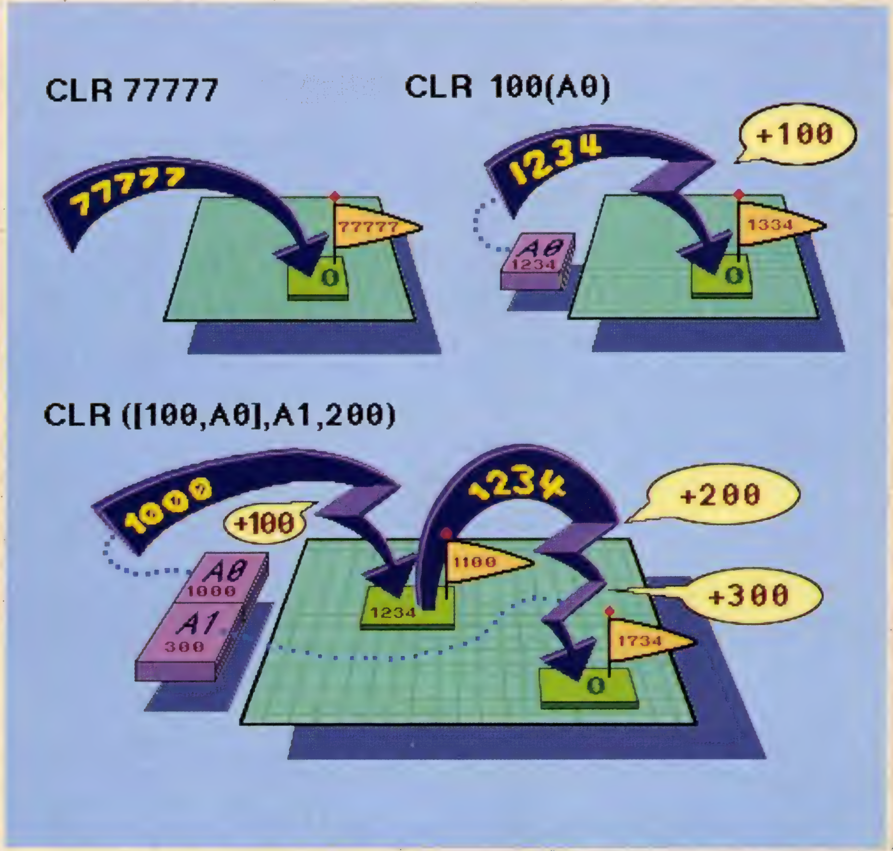


図4　左上はダイレクトモードによるアドレス77777のクリア（アドレス自体には意味はない）．右上はインデックスモードによるもの．汎用レジスタであるA0レジスタに記憶されていたアドレス1234にオペランドで指定したオフセットである＋100を加えた，アドレス1334をクリアしている．下は，非常に複雑なアドレスモードであるメモリ間接参照．A0からインデックスモードで参照したアドレス1100の内容であるアドレス1234を基に，さらに第2のインデックスレジスタとしてA1を使い，最終的にアドレス1734をクリアしている．

命令の実行

前項で述べた１つの命令も，実は命令フェッチ，デコード，などの各段階に分けて実行されている．これらの処理には当然相応の時間がかかるが，CPU内部では独立したハードウェアで実行されるため，それぞれのハードウェアが並行して動作するように工夫すると，CPU全体としての高速化ができる．

命令実行のステージ

コンピュータ内部では，クロックと呼ばれる周期的なパルスのON／OFFのタイミングに合わせて順次処理が進行していく．１つの命令の処理も，すべてが１クロックで行なわれるわけではなく，いくつかのステージに分割して行なわれる．

命令フェッチ　PCに書いてあるアドレスのメモリから，命令を取り出してくるステージ

デコード　読まれた命令を解読し，必要な制御信号を用意するステージ

オペランドフェッチ　命令実行に必要な入力データをレジスタやメモリから読み出し，ALUに供給するステージ

メモリ読み出し　メモリの内容を読み出すステージ（オペランドフェッチでメモリアクセスが必要な場合）

演算　ALUで実際に演算が行なわれるステージ

オペランドストア　ALUなどからの命令実行の結果の出力をレジスタやメモリに書き込むステージ

メモリ書き戻し　メモリにデータを書き戻すステージ（オペランドストアでメモリアクセスが必要な場合）

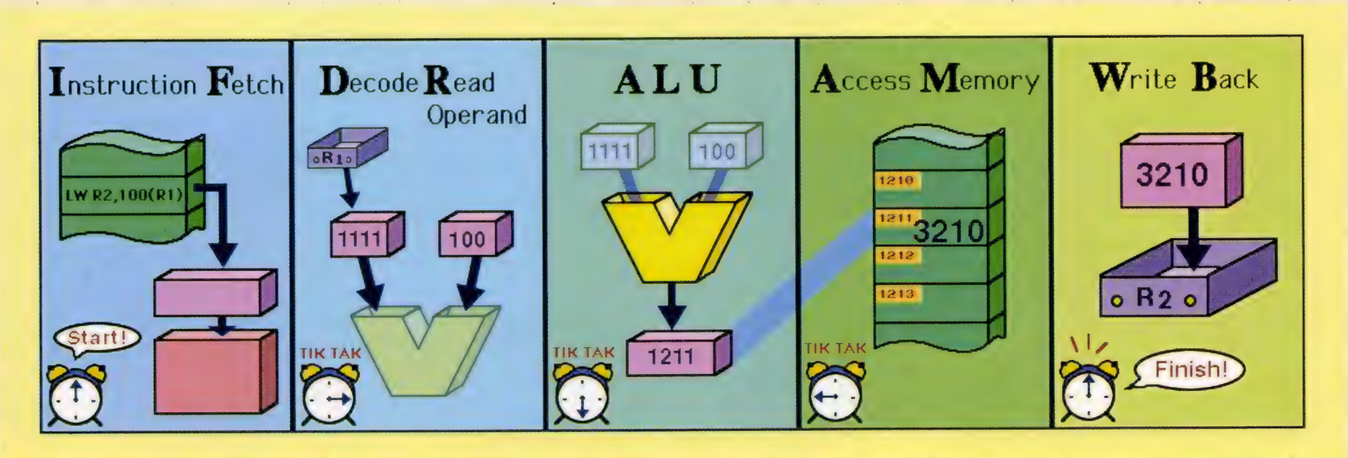
これらのステージのうち，演算ステージなどは，複雑な演算の場合多くのクロック周期を必要とする．メモリアクセスも，一般に多くのクロック周期を必要とする遅い操作である．複雑なアドレスモードを利用した場合には，オペランドフェッチやオペランドストアはいくつかのメモリアクセスを含む，長く複雑なものとなる．また複雑な複合命令では，オペランドフェッチや演算のステージを複数回必要とするものもある．図５に，メモリ読み出しの実行のようすをステージに分けて示す．

マイクロプログラミング

一般に命令の実行の制御には，多くの機能ユニットを正確なタイミングで，きめ細かく制御しなければならない．同じオペコードの命令でも，アドレスモードによって必要なステージ数は変化するし，そもそも，各ステージにかかるクロック数も一定とは限らない．つまり，複雑な命令の実行の制御は，極めて複雑なものとなる．こういった複雑な制御には，ハードウェアで直接組んだ制御回路では，間違いがおきやすいし，設計変更があった場合の回路の変更が大変である．このような複雑な制御には，ソフトウェアのほうが向いている．そこで，コンピュータ自体の制御信号を発生するため，内部に小さなコンピュータを組み込み，そのプログラムによって全体の制御を行なうという考え方がある．これが，マイクロプログラミングである．

マイクロプログラムは，CPU内部の高速のROMやRAMに格納され，１クロックサイクルに１命令が実行できるように工夫されている．マイクロプログラムの方式には，垂直型と水平型の２種類がある．垂直型では，マイクロ命令は通常の命令と同様デコードされて制御信号となるため，命令は短くなるが同時に制御できるユニットの数は制限されるし，命令のデコードに多少時間がかかる．一方水平型では各ユニットの制御信号をすべて命令中に直接持ち各ユニットを完全に個別に制御できるため，高度な並行動作を記述できるが，命令は数十bitから数百bitの長さになる．水平型ではCPU内部の限られた記憶容量にはあまり多数の命令を

図５　アドレスをインデックスレジスタの値をもとに計算し，メモリからのロードを行なう．これだけの処理でも命令の読み込み，命令の解釈，アクセスすべきアドレスの決定，実際のメモリの読み出し，レジスタへの書き戻しと，多くのステージに分けて行なわれる．



PCで指定されたメモリから，次に実行すべき命令であるロード命令を，インストラクションレジスタに読み込む．

命令をデコードし，そこで指示されたオペランドを，ゼネラルレジスタから読み込むなどして用意する．

ALUは，オペランドとして与えられたアドレスに演算をほどこし，最終的な処理の対象となるメモリアドレスを求める．

前のステージの実行結果として得られたアドレスを指定して，実際にメモリからデータを読み込む．

メモリをアクセスした結果を，オペランドで指定されていたゼネラルレジスタに書き戻し，処理が終了する．

持てない．実際のコンピュータでは，垂直型か，垂直型に水平型を混合したような形式がよく用いられる．

マイクロプログラムの積極的な利用により，非常に複雑な命令の作成が手軽に行なえるようになった．たとえば，Pascalのような高級言語を直接実行するコンピュータも作成可能である．また，RAMにマイクロプログラムを格納することにより，実行時に動的に命令を変化させることも可能となった．

パイプライン

コンピュータの命令の実行は，クロックによって区切られたいくつかのステージに分けて行なわれる．簡単なコンピュータでは，1つの命令の全ステージの実行が終わってから次の命令の実行にとりかかる．しかし，高速な命令の実行のためには，それでは無駄が多いので，パイプラインという技術により，複数の命令を同時に実行する．

命令実行の各ステージにおいては，それぞれ特定のハードウェアユニットが使用される．たとえば，演算ステージでは，ALUが使用される．しかし，その他のステージを実行中にはALUは使用する必要がない．であるから，先に実行が開始された命令がすでに使ってしまったユニットや，まったく使う必要のないユニットについては，その命令の実行完了を待たずに，次の命令が使用を始められる．また，処理に時間がかかるユニットは，前半の処理と後半の処理の部分に分割すれば，前半と，後半を担当する部分を別々の命令で同時に利用できる．このようにすると，各ユニットは水道パイプに連続して水が流れるように連続的に利用されるので，この方法をパイプライン処理と呼ぶ．

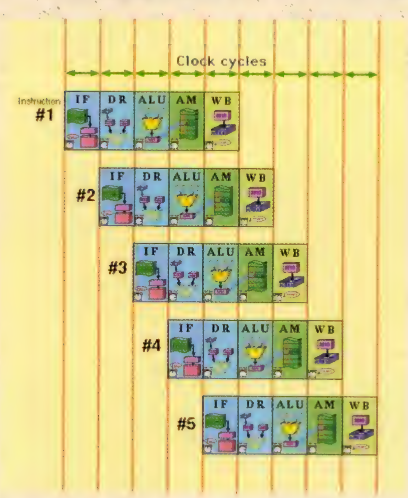


図6　最初の命令の1ステージ分の処理が終わり次のステージに入ったところで，次の命令の処理を開始すれば，パイプラインとなる．

パイプライン処理が完全にうまく動作すると，1クロックあたり1命令実行できるが，後の命令が使いたいユニットを前の命令がまだ使用中であったり，前の命令の実行結果が後の命令で必要であったりすると，後の命令は，実行を待たなければならない．これをパイプラインの乱れと呼び，実行速度の低下を招く．ジャンプ命令も，次に実行すべき命令が遠くにあったり，条件ジャンプの場合ジャンプするかどうか分からなかったりするので，パイプライン処理の効率を低下させる要素である．

割り込みとコンテキストスイッチ

あるプログラムの実行中に，プログラムの本来の動作とは無関係な事態が発生し，CPUがもとのプログラムの処理を中断する必要が起きることがある．これを割り込みという．

割り込みにより，CPUはある仕事の待ち時間を活用して，他の仕事を行なうことができる．たとえば，ディスクに対する出力をいくつか行なう場合に，1つの転送を設定してしまえば，ディスクのヘッドの移動や，シリンダの回転待ちの間は何もすることがない．そこで，この待ちの間は，CPUは他の処理をすることにする．しかし割り込みがない場合は，CPUは転送が終了したかどうか頻繁にチェックしなければ，次の転送をいつ始めてよいか分からない．そこで，転送終了時に割り込みがかかるようにしておけば，終了が通知されるので次の転送が連続して行なえ，ディスクの速度を有効利用できる．

割り込みは，0による除算や，存在しないメモリアドレスの利用のような，正常なプログラムの動作では起こりえない事態に対しても発生する．このような異常事態では，通常プログラムの実行は中断されるが，適切な対処により実行が継続できる場合もあるからである．

割り込みは，ハードウェアやマイクロプログラムによって検出され，サブルーチンコールに似た方法で，割り込み処理プログラムが呼ばれる．割り込み処理プログラムでは，まず，もともとの処理が中断した場所から再開できるよう，これまでの実行環境を保存する．この情報をコンテキストといい，PCやゼネラルレジスタの値などからなる．割り込み処理プログラムが終了すると，コンテキストが復元され，もとのプログラムの実行を再開する．

時計から短い時間間隔で割り込みを発生し，複数のコンテキストを高速に切り替えて2つ以上のプログラムを走らせることによって，複数の人間が同時に1台のコンピュータを利用するTSS（Time Sharing System）というシステムも一般的である．多人数でのTSS処理などのため割り込みには素早く応答できることが望ましいが，多数のレジスタがあるとコンテキストの保存／復元に余分な時間がかかる．実行に長い時間がかかる命令の途中での割り込みの場合，その命令の実行を中断して割り込み処理を行なう場合もある．この場合，コンテキストとしてその命令がどこまで進んでいたかという情報も必要になる．

パイプライン処理においては，割り込みはやっかいな問題である．パイプライン処理の途中で割り込みが起きた場合，同時に実行されている複数の命令を中断するわけだが，命令実行の結果としての割り込みであった場合は，その命令以降の命令の実行は本来行なわれていないはずなので，並行して実行してしまった部分の影響を取り消さなければならない．また，割り込みはパイプラインを乱すため，実行速度を低下させる可能性もある．

RISCを支える基礎技術

コンピュータの高速化にはフォンノイマンボトルネックを流れるデータ量を減少させることが重要である．そのためには，CPUが記憶できる情報量を増加させるとよい．記憶階層におけるキャッシュメモリ，仮想記憶システムでのTLB，大量のレジスタファイルなどはすべてそのためのものである．

記憶階層

一般に記憶装置に望ましい性質として，大容量の記憶が可能であることと，高速のアクセスが可能であることがある．しかし，この2つの要求は相反するものであり，同じコストでは高速低容量の記憶装置と低速大容量の記憶装置が存在する．コストの問題を無視するとしても，本来高速である記憶装置も，大容量にするべく多数並べると，配線の容量やアドレスデコードの段数の増加のため，速度は低下してしまうという問題がある．

そこで，記憶の階層化の考えにより，何種類かの記憶装置をうまく組み合わせて利用する．まず，CPUに近い部分には高速で小容量のレジスタを配置する．一方最もCPUから遠い部分には，磁気ディスクを置く．途中には極めて高速のキャッシュメモリと，まあまあの速度のメインメモリが置かれることになる．

とはいえ，利用者がこのような記憶階層をいちいち意識して利用しなければならないのでは煩雑なので，キャッシュ，メインメモリ，磁気ディスクはすべて通常のメモリアクセスと同様な統一的インターフェイスで利用できるようにする．

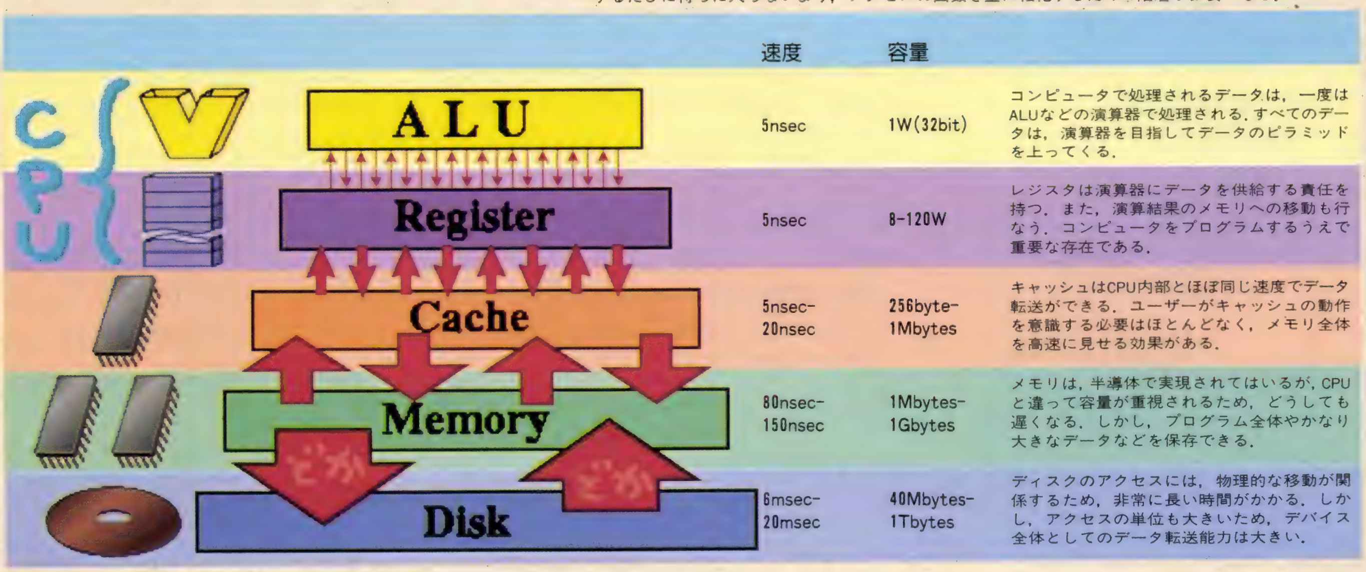
キャッシュ

キャッシュ（Cache memory）とは，CPUとメインメモリとの間に存在し，メインメモリの一部のデータのコピーを保持する小さくて高速の記憶装置である．メインメモリは通常MOS（Metal Oxide Semiconductor）のDRAM（Dynamic Random Access Memory）が利用されるが，最近ではキャッシュ用に高速で比較的大容量のSRAM（Static Random Access Memory）が利用できるようになってきた．

近年，CPUの速度は2～3年で倍増しているが，メインメモリの速度はそれほど速くなってきてはいない．CPUは1クロックに1命令実行できても，メインメモリをアクセスするには，指定されたアドレスを見つけるために数クロックから十数クロックの時間が必要となる．しかし，一度番地が見つかると，メモリのインターリーブ（デバイスにアドレスを割り付ける際，連続したアドレスはすべて異なるデバイスに割り付け，連続した領域へのアクセスを並列的にできるようにする手法）により，その付近のデータの連続した転送は高速に行なえる．一方，メモリを利用するプログラムの多くは，短い時間には同じアドレスや連続したアドレスのメモリばかり利用する傾向がある．そこで，メインメモリとCPUの間に，高速で小容量のメモリであるキャッシュを置き，メインメモリのデータの一部のコピーを持たせることになる．

キャッシュを持つコンピュータでは，メモリがアクセスされた場合，まずキャッシュ内部にコピーがないかどうかがチェックされる．もしコピーがなければ，時間はかかるがメインメモリがアクセスされ，その値が使われる．同時に，アクセスされたデータのコピーがキャッシュの空いた部分に置かれる（空きがない場合，古いコピーが適当に捨てられ空きが作られる）．またアクセスされたアドレスだけでなく，その付近の何byteかもまとめて読まれる．同じアドレスや続いたアドレスが続けて読まれた場合には，キャッシュにコピーがあるので，メインメモリをアクセスする必要がない．メモリへの書き込みの場合も，キャッシュのコピーにも書き込みが反映されるため，書いた結果をすぐ読んでも，値が食い違うこ

図7　コンピュータ内部にはデータのピラミッドがある．ALUからディスクまで，データの量が増えるに従って速度が遅くなる．しかし，データを必要とするのは高速なALUである．ALUがデータを必要とするたびに待ちに入らないよう，アクセスの回数を量に転化するための階層が必要になる．



とはない．

キャッシュはまた，しばしばデータ用と命令用とに分けて使われる．これは，普通のデータと命令では，メモリの利用方法が異なるためである．命令は，命令を実行中に書き換えるなどの特殊な使い方をしないかぎり，読み出されるだけである．したがって，命令用のキャッシュの内容がメモリから読み込まれた後で変化することはあまりない．であるから，キャッシュの内容をメモリに反映させるための機構が単純になり，簡単な構造でキャッシュが実現できる．また，ジャンプ命令以外では連続した領域が順番に利用されるという性質を持つため，1回のメモリアクセスで大量のメモリを読み込んでも無駄になりにくい．したがって高速なキャッシュが組みやすい．また，データとして利用するアドレスの範囲と命令として実行するアドレスの範囲は通常別々なので，2つのキャッシュは独立に動作できる．

仮想記憶

仮想記憶という技術により，磁気ディスクも，メインメモリの一部であるかのようにアクセスできる．仮想記憶システムでは，メモリはページと呼ばれる数Kバイトの単位で管理され，プログラム中で利用される仮想アドレスから，実際のメインメモリのアドレスである実アドレスへの変換がなされる．仮想アドレスによっては，その内容が磁気ディスクにしかなく，対応するメインメモリが割り当てられていない場合がある．この場合自動的に磁気ディスクからメインメモリへの転送が行なわれる．

仮想アドレスから実アドレスへの変換は，原理的には巨大なテーブルをひくことによって行なわれるが，これには数回のメインメモリへのアクセスが必要となる．仮想アドレスを利用するたびにこのような手法でアドレスの変換を行なっていては，メモリアクセスの手間は数倍に増えることになる．これを防ぐためには，最近のアドレス変換の結果をどこかに覚えておけばよい．このために存在するのがTLB（Translation Lookaside Buffer）である．アドレスの変換の際には，まずTLBを調べ，その仮想アドレスが登録されていれば，それを使う．

仮想記憶を持ったコンピュータでキャッシュメモリをアクセスする場合，仮想アドレスでアクセスするか実アドレスでアクセスするかの2つの方式があり，それぞれ一長一短がある．実アドレスでアクセスする場合には，キャッシュアクセスの前にアドレス変換を行なう必要がある．この場合TLBによる高速の変換が行なえないと，キャッシュを利用する意味がなくなる．

レジスタファイル

大量のレジスタファイルもフォンノイマンボトルネックを通るデータを減少させる便法の1つである．レジスタファイルはキャッシュメモリとは異なり，その利用の方法は，完全にプログラムで指定される．大量のレジスタファイルを効率よく利用するには，人手によるレジスタの割り付けでは限界があり，高度なコンパイラによって割り付けるか，レジスタウィンドウと呼ばれる方法を利用することになる．

MOSとバイポーラ

現在のLSIには，基本的にMOS回路によるものとバイポーラ回路によるものの2種類がある．

一般にMOSと呼ばれる回路はチップ表面のごく薄い層をFET（Field Effect Transistor，電界効果トランジスタ）として使い，構成される．半導体がp形のFETと半導体がn形のFETを使ったものを，それぞれpMOS，nMOSと呼ぶ．nMOS単体でも使われるが，この2つは電気的に相補的である．このため両方の組み合わせによるCMOS素子は，定常状態では電流がほとんど流れないという特性を持つため，多く使われる．スイッチ動作に必要な素子がCMOSでも2つですむため，基本的なゲートの構成に必要とされる素子の数は，比較的少ない．

このような特性のため，MOSの回路は，高密度，特に動作クロックが低い場合に低消費電力，比較的低速という特性を持ち，高密度が要求されるメモリやプロセッサに使われている．

バイポーラ回路は，チップ面に対して垂直に作られるトランジスタによって構成される．電流が電圧の指数関数に比例して連続的な変化をするという特性を持つ．これを飽和状態で使用するのがTTL（Transistor Transistor Logic）回路や，IIL（Integrated Injection Logic，マルチコレクタのトランジスタと通常のトランジスタを融合したもの）回路である．しかし飽和状態ではスイッチ特性に遅れが生じる．非飽和状態（電流が電圧によって連続的に変化する状態）で使うのが，ECL（Emitter Coupled Logic）回路である．ECL回路は定常電流を2つのトランジスタに振り分けることによってスイッチングを行なうため，電力消費は大きい．また，ECL回路はゲート1つを構成するために必要な素子の数も多くなる．

TTL回路は，外部へのドライブ能力が大きいことなどにより標準論理ICなどに使われているが，集積度を上げにくく，また低速なため，VLSIにはあまり使われていない．ECL回路は，たとえ低集積度になっても高速性の必要な回路に多く使われている．

メモリの構成

メモリは読み出しと書き込みの両方が可能なRAMと，読み出しのみが可能なROMに大別される．いずれの場合も1bitの記憶を担当するメモリセルを2次元配列として並べ，行と列のアドレスを与えて読み出したり書き込んだりするようになっている．メモリセルは集積度を上げるために微弱な信号しか出せなくなっているので，出力を増幅して出力するようになっている．

RAMは，自力では一定時間しか記憶の保持できないダイナミックRAMとメモリセル自身に長時間の記憶保持能力のあるスタティックRAMに分けられる．

ダイナミックRAMのメモリセルはMOS FET 1個と，コンデンサ1個で構成され，記憶はコンデンサへの充電によって正となる．読み出しは，コンデンサの電荷を取り出すことによって行なわれる．このため，読み出しによってデータは破壊される．また，電荷の自然放電によって，データは数msec．で消滅する．このため，一定時間おき，また読み出し時に各メモリセルをリフレッシュする必要がある．しかし，メモリセルの構造が単純なため，集積度は上げやすい．

スタティックRAMのメモリセルは，フリップフロップにより記憶を保持するため，比較的多くのトランジスタを必要とする．そのため，ダイナミックRAMと比較すると集積度が低くなる．しかし，バッテリバックアップなどにより，CPUと独立して記憶を保持できる．また，ダイナミックRAMに比べて，高速のメモリが作れる．中でも高速のメモリには，ECLの回路が使われることが多い．

ROMには製造時に固定的にデータの書き込まれたマスクROMと，電気的に書き込み可能なヒューズROM，紫外線照射によって消去できるEPROM，さらに電気的に消去できるEEPROMなどがある．ROMは一般にダイナミックRAMと同程度の実装密度のものがあるが，速度は，ダイナミックRAMに比べても低速である．このため，特に高速性が必要とされる場合は，いったんRAMにデータをコピーしてから使うことがある．

RISCとは何か

複雑な処理の高速化には，専用ハードウェアとそのための専用命令を用意するCISC的アプローチと，単純な命令の組み合わせを高速に実行するRISC的アプローチがある．これまで述べてきたコンピュータの高速化のための技法には適用範囲に限界があり，技術の進歩に伴って再評価する必要がある．

32bit命令キャッシュ

CISCとRISCとの表面的な違いは，命令体系にあるわけだが，これに直接的に影響しているのは，32bitの命令キャッシュの存在である．

まず，32bit CPUでは命令の長さが十分長くとれるため，命令体系に工夫を凝らさなくとも，十分1語でいろいろなことが表現できる．そこで，可変長命令は不要となる．

また，命令キャッシュの存在により1クロックに1命令取り出せ，パイプラインにより1クロックに1命令実行できる．そこで，マイクロプログラムは速度的に通常の命令と同じ速度でしか実行できない．まして，命令の長さが32bitもあれば，もはや垂直型マイクロプログラムを使う余地はない．

ほとんど順次に行なわれる命令取り出しが1クロックで行なえるのに比べて，ランダムなデータのアクセスには，長い時間がかかる．1命令中でメモリアクセスを2度も必要とするようなアドレスモードを指定して，データアクセスが終わるのを待つくらいなら，単純なアドレスモードを使った命令を複数組み合わせたほうが，はるかに短い時間で処理できる．

速度の追求

このような，これまでのコンピュータの基礎的な技術の意義の変化をもとに，どうすれば高速なコンピュータが構成できるかを考えてみよう．

ある特定の仕事を行なうのに必要な時間は，

仕事にかかる時間 ＝ C ＊ T ＊ I
C：命令あたりのクロック数
T：クロックの長さ
I：仕事あたりの命令数

となる．そこで，C，T，Iをそれぞれ減少させることにより，全体の速度を向上させることができる．

命令あたりのクロック数

ある命令を実行するのに必要な時間を短縮するには，パイプライン技術により，複数の命令の実行を並列に行なうのが効果的である．ただし，パイプラインは，なるべく乱れが少なく動作しなければならない．そこで，**図8**と**図9**を見比べてほしい．**図9**は，CISCによるパイプラインの動作の例である．各命令は機能が高く，アドレスモードなども存在する．そのため，1つの命令がいろいろなユニットを途中で利用し，全体としてパイプラインが効率よく動作せず無駄ができる．

このような事態を防ぐためには，命令の並べ方を工夫して，実行途中でのユニットの利用がぶつからないようにすればよい．

そのための最も簡単な方法は，すべての命令の実行のパイプラインの長さを同じにし，各段階で異なるユニットを利用することである．そうすれば，異なる時刻に開始された命令同士が，同じ時刻に同じユニットを必要とすることはなくなる．**図8**の例では，すべての命令は，フェッチ，デコード，演算，メモリアクセス，ライトバックのまったく同じ5ステージのパイプラインからなり，どのステージも1クロックで終了するため，パイプラインは，まったく乱れなく動作する．ただし，命令フェッチを1クロックで行なうためには，命令の長さは固定長でなければならない．

CISCにおいても，複雑な制御回路を工夫することにより，簡単な命令の実行だけなら，パイプラインを乱さず1命令1クロックで行なうことは可能であり，汎用大型コンピュータなどは実際そうなっている．しかしそのような場合，複雑な命令を使わずに単純な命令だけを組み合わせたほうが処理が高速になるので，結局のところ複雑な命令には存在意義がない．

しかし命令によっては，すべてのステージが1クロックでは終了させられるわ

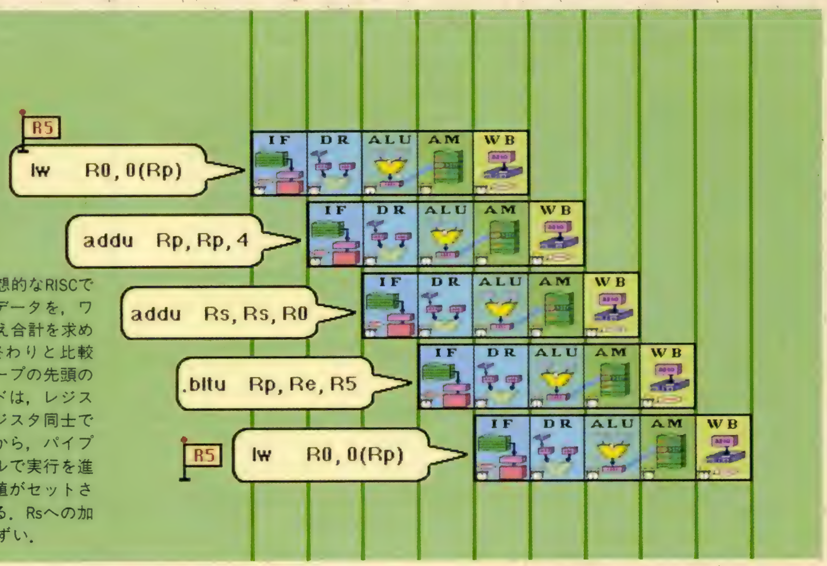


図8　5ステージのパイプラインを持つ仮想的なRISCでデータの総和を求める．ポインタRpの指すデータを，ワーク用のレジスタRwにロードしてはRsに加え合計を求める．Rpはインクリメントして，データの終わりと比較し，まだデータがあるようならR5の指すループの先頭のアドレスにブランチする．命令のオペランドは，レジスタの3個程度で短く，ほとんどの処理はレジスタ同士で行なわれ，長い時間かかるステージもないから，パイプラインはよどみなく流れ，1命令1サイクルで実行を進められる．一見当たり前の処理だが，Rwに値がセットされるのは，実は次の命令の先頭より後である．Rsへの加算命令は，Rwへのロード命令の直後ではまずい．

けではない．メモリからの読み込みは，たとえキャッシュがうまく働いても時間のかかるものだし，乗算／除算などの複雑な演算もすぐには終わらない．これらの遅延があるとき，処理の終了を待つことなく次の命令を実行してしまうことができる．次の命令が，前の命令の結果を利用することさえなければそれで構わないわけである．後でその結果が必要になった場合に，初めて待ちが入る．そして，よいコンパイラを利用すれば，たいていの待ち時間は有効な仕事に使える．ただし，このように命令の終了がいつなのかはっきりしない命令体系では，アセンブラ言語でのプログラミングは，非常に困難になる．

クロックの長さ

クロックの長さは，パイプラインの各段階を実行するのに必要な時間で決まる．**図8**の例では，各ステージのうち最も遅い部分の時間で決まる．

命令キャッシュがデータキャッシュと分離されたコンピュータでは，フェッチは高速に行なえるので，そのための時間は問題にならない．

デコードを高速に行なうためには，まず命令の形式が統一的であることが望ましい．命令がさまざまな形式を持てると，まずその命令がどの形式に属するものかを判断してから，残りの部分を解析しなければならないからである．

演算に関しては1サイクルで終了できれば，後に影響を残さず一番望ましい．整数の加減算などは，1クロックで行なえるようにクロックを設定する．しかし，浮動小数点演算のようにどうしても低速なものはその終了を待たずに次の命令を実行して，クロックの速度は落とさない．

結果のストアは，レジスタに対しては高速に行なえるし，メモリに対しても，いちいち終了を待つ必要はないので，特に時間はかからない．

仕事あたりの命令数

ある仕事をするのに必要な命令数を減らすには，その仕事専用の命令を用意すればよいとは，誰もが考えることである．CISCは，まさにそのような考えに基づいて作成されたわけであるが，実際に使われている状態を調べてみると，意外なことが分かった．まず，複雑で高機能な命令は，高級言語に直接対応する機能がない限り，コンパイラがサポートしないので，使われにくい．またサポートされていたとしても，使われる機会自体そもそも少ない．なぜなら，高機能な命令は，それだけ専用化しており，少しでも異なった要求には対応できないからである．

そこでコンパイラは，多くの場合最も基本的な命令を組み合わせて必要な処理を実現することになる．特に，実行に複数クロックや複数回のメモリアクセスが必要で，かつ単純な操作の組み合わせで効率よく表現できる命令は，命令の取り出しの手間が無視できるため用意する必要がない．

したがって高機能な命令としては，メモリアクセスを伴わず，専用ハードウェアと専用命令の組み合わせにより飛躍的に速度が向上する，浮動小数点演算程度以外は必要がなくなる．

RISCの誕生

高速な処理の実現のためには，

命令長固定
簡潔で規則的な命令体系
均一なパイプライン
命令キャッシュとデータキャッシュの分離
多数のゼネラルレジスタ

が重要で，これがRISCの本質である．

この中で，簡潔な命令体系を持つことには，ほかにも利点がある．1つは設計の容易さである．簡潔な命令を持つCPUを設計するのは，複雑な命令を持つCPUを設計することに比べてはるかに簡単である．したがって，開発にかかる時間が短くて済み，そのぶん最新の（すなわちより高速の）半導体技術をもとに設計できる．また，設計の誤りも起こりにくい．最後に，命令が簡潔であれば，高度に最適化されたコンパイラの作成もより単純に行なえる．

このように，命令体系の簡潔さが，コンピュータの性能にとって重要であることに着目し，RISCという名前が使われているのである．

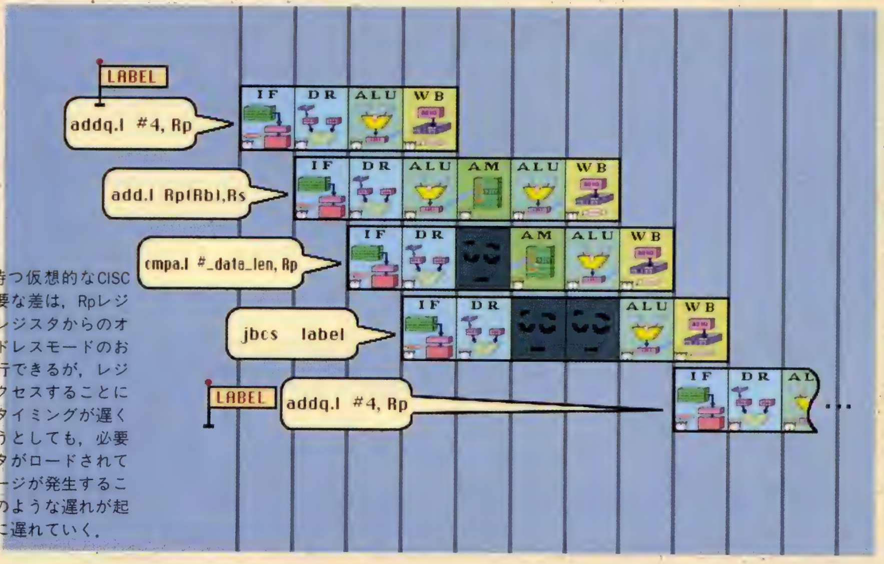


図9　5ステージのパイプラインを持つ仮想的なCISCで，図7と同様の処理をしている．重要な差は，Rpレジスタに直接ポインタを使わず，ベースレジスタからのオフセットを入れていること．複雑なアドレスモードのおかげで，この程度の処理は1命令で実行できるが，レジスタ同士の加算をしてからメモリをアクセスすることになるため，メモリアクセスを発行するタイミングが遅くなる．その結果，次の命令を実行しようとしても，必要な演算器が空いていなかったり，データがロードされていなかったりして，何もできないステージが発生することになる．処理の途中で，いったんこのような遅れが起きると，全体のスケジュールが連鎖的に遅れていく．

RISCコンパイラの現状

コンピュータ技術の発展によって，特に速度が要求されるような応用やＯＳにも，アセンブラ言語ではなく高級言語が使われるようになった．このため，コンパイラ技術によって，ハードウェアをうまく活かしたコードを生成することが非常に重要になってきた．そのために必要な高速化技術を例示する．

インライン展開

インライン展開とは，本来サブルーチン呼び出しとなる部分を，サブルーチンの本体と置き換えて直接実行する最適化である．たとえば，

```
max(i,j)
int i,j;
{
    return(i > j)? i: j;
}
    ...
    for(i = 0; i < N; i++)
      a[i]= max(a[i], b[i]);
```

の関数maxをインライン展開すると，

```
for(i = 0; i < N; i++){
    s = a[i] ;
    t = b[i] ;
     a[i]= (s > t)? s: t;
}
```

となる．展開により，サブルーチン呼び出しのために必要であったスタックポインタの操作や，パイプラインを乱すもとになるサブルーチンへのジャンプなどが不要になる．サブルーチンが長い場合にはあまり効果はないが，ループの中で平方根を求めるサブルーチンを繰り返し呼んでいる場合などには非常に有効である．

レジスタ割り当ての最適化

レジスタは，最も高速な記憶場所であるので，その効率的な利用は，性能向上に大きく影響する．Ｃ言語では，昔はコンパイラがそれほど複雑でなかったため，利用者がどの変数をどのようにレジスタに割り付けるかをregister宣言という構文により指定でき，特に指定しない変数はメモリに割り当てられるようになっていた．Ｃ言語は，もともとレジスタが８個しかなかったPDP-11というミニコンピュータ用に設計されており，自由に使えるレジスタが少ないことを前提にしている．

しかし，RISCでは多数のレジスタが存在する．これらの最適な利用を人間が考えるのは難しい．また，CPUによって最適なレジスタの割り当ての変化まで考慮したプログラムをしなければならない．これでは大変なので，RISCのコンパイラは，特にregister宣言をしない変数に対しても自動的にレジスタの割り当てを行なってくれる．

レジスタ割り当てのさらに進んだ形として，グローバルレジスタ割り当てがある．サブルーチンを呼ぶ場合，レジスタの利用に関しては約束ごとがある．あるレジスタは引数を渡すのに利用するし，あるレジスタは内容の保存が保証されるので，サブルーチンを呼び出す側が内容を保存する必要はない．また，あるレジスタは，各サブルーチンで完全に作業用に利用して破壊してよい．通常は，どこからどの関数が呼び出されるか不明なので，この約束ごとは全体で統一されていなければならない．ところが，プログラム全体を一度にコンパイルするなら，コンパイラがサブルーチンの呼び出し関係を完全に把握できるので，**図10**のようにサブルーチンの構成に応じてレジスタ使用のより高度な最適化をし，レジスタ保存の手間を省くことができる．

ループアンローリング

短いループを回る場合，ループが終了したかどうかの判断や，パイプラインを乱すことになるジャンプ命令は，性能低下の原因となる．そこで，ループがたとえば４回以上実行されると分かっている場合，ループの中身を４回書き並べて，終了判定とジャンプの手間を1/4にすることができる．この最適化がループアンローリングである．たとえば，

```
for(i = 0; i < 100; i++)
    *p++ = *q++;
```

のようなループを４重にアンローリングしてみると，

```
for(i = 0; i < 25; i++)
```

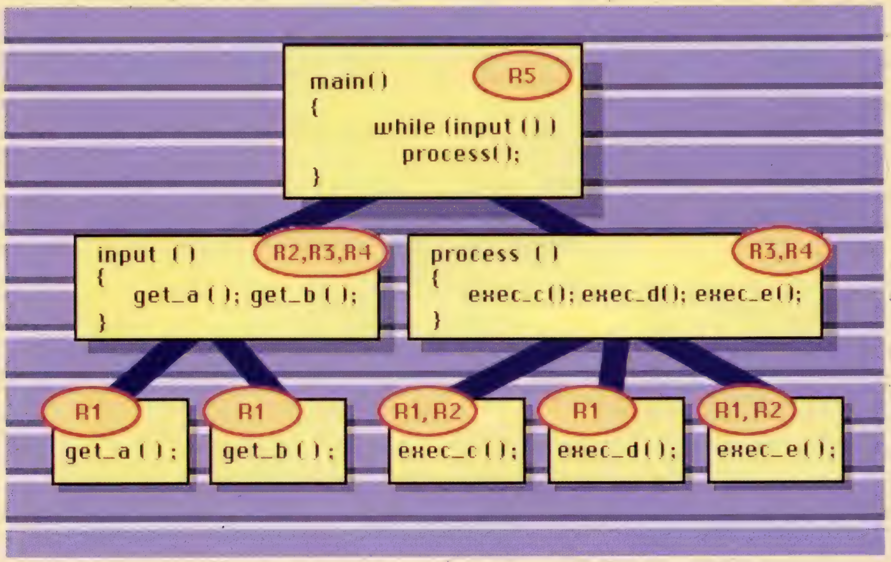


図10　本来なら他のサブルーチンを呼ぶ場合には，レジスタを保存しなければいけないが，作業用に使うレジスタを，うまくサブルーチンごとに割り振れば，レジスタの保存を省略することができる．サブルーチンの呼び出し関係が，図のようになっているとする．inputからはprocess，あるいはexec系のルーチンが呼ばれることはないので，inputおよび，get系のルーチン内部では，mainで使われていないレジスタは自由に使ってよい．そこでレジスタR1〜R4を作業用に使うとする．さてinput内部でR2〜R4を使うとすると，inputから呼ばれるget系のサブルーチンではR1を保存せずに使うことができる．processについても同様の割り付けが行なわれる．

```
{
        *p++ = *p++;
        *p++ = *p++;
        *p++ = *p++;
        *p++ = *p++;
}
```

となる．ロード命令を先にまとめるなど，命令を並べ替えることにより，さらに高速化することも期待できる．

遅延分岐

一般的なRISCにおいてはパイプライン処理の結果，ジャンプ命令が実行される場合，すでにジャンプ命令の次にある命令も実行態勢にはいっている．これは，ジャンプが起きれば本来実行されないはずの命令である．パイプライン処理を工夫すれば，ジャンプ命令の実行後に，そのよけいな命令の実行を取り消すことも不可能ではないが，処理が複雑になるし，その結果性能が向上するわけでもない．そこで，RISCでは一般に，ジャンプ命令の次の命令は，ジャンプが起きようが起きまいが実行してしまうこととする．このような実行順序を遅延分岐(Delayed Branch)と呼び，ジャンプ命令の次の命令を遅延スロットと呼ぶ．遅延スロットに適切な命令を持ってくるのが遅延分岐に関する最適化である．

まず，**図11**にあるように，遅延スロットには最初NOP命令が挿入されているとする．**図12**のように，このスロットにジャンプ命令の直前の命令を持ってくることができればそれでいい．場合によっては，ジャンプ先の命令をこのスロットに持ってくることもある．この場合，ジャンプが無条件ジャンプであるか，ジャンプが起きなかった場合でも遅延スロットの命令が無害でなければならない．

リオーダリング

パイプライン処理を行なう場合，重要なのはパイプラインを止めることなく流し続けることである．パイプラインの止まる原因としては，先の命令が使っているレジスタや演算ユニットなどの資源を後の命令も使いたいが，まだ先の命令が使い終わってない場合がある．RISCでは，命令の実行は極力1クロックで終わるようにデザインされているはずだが，一部の命令はどうしても複数クロックサイクルを必要とする．メモリからデータを読み込む命令や，整数の乗算／除算，浮動小数点演算などがこれにあたり，実行に待ちが生じることになる．そこで，複数サイクルかかる命令をなるべく早く実行するように実行順序を変更すると，パイプラインが前の処理の結果待ちで滞ることなく円滑に処理が進む．このような最適化が命令のリオーダリングである．

ストレングスリダクション

次のようなループを考える．

```
for(i=0; i<N; i++)
    a[i*3]=0;
```

このときループ内部では，本来 i に3を掛けるわけだが，乗算は，あまり速い操作ではない．a はどうせ3刻みで順にアクセスされるだけなので，プログラムを

```
p=a;
for(i=0; i<N; i++)  {
    *p=0;
    p+=3;
}
```

のように変換すれば，乗算は必要でなくなり，加算だけでよい．このようにループ内部で本来必要だったはずの演算を，より簡単な演算の累積に置き換える最適化の手法を，ストレングスリダクションという．

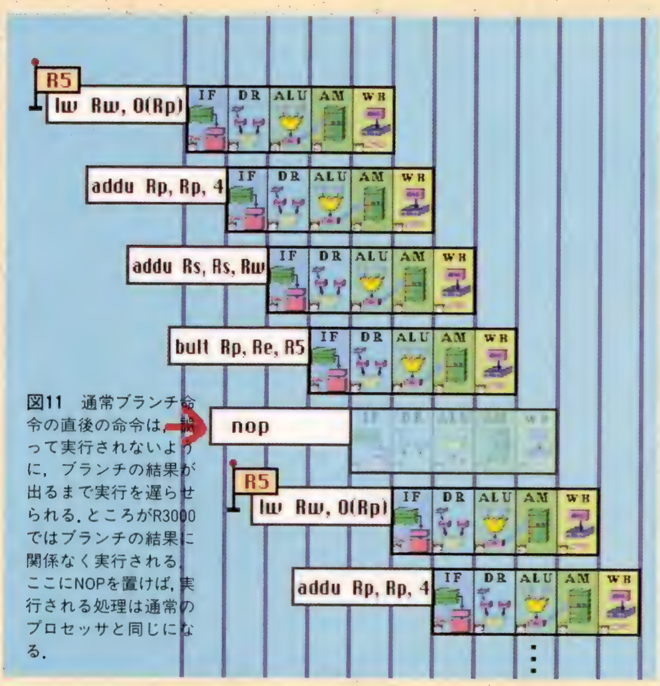


図11　通常ブランチ命令の直後の命令は，誤って実行されないように，ブランチの結果が出るまで実行を遅らせられる．ところがR3000ではブランチの結果に関係なく実行される．ここにNOPを置けば，実行される処理は通常のプロセッサと同じになる．

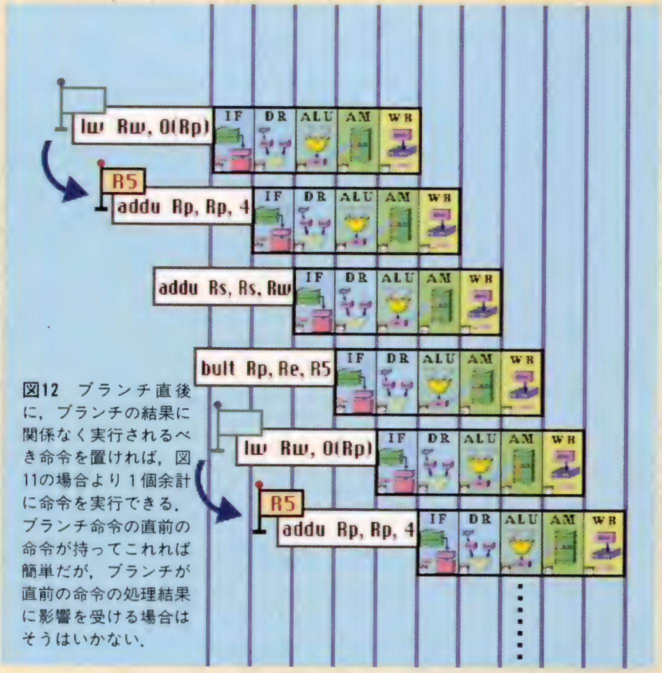


図12　ブランチ直後に，ブランチの結果に関係なく実行されるべき命令を置ければ，図11の場合より1個余計に命令を実行できる．ブランチ命令の直前の命令が持ってこれれば簡単だが，ブランチが直前の命令の処理結果に影響を受ける場合はそうはいかない．

RISCの将来

すでにコンピュータの発展の主流となったRISCは，この先どう発展していくだろう．すでに1命令1サイクルを達成した現在，発展の方向としては，同時に処理できる量を多くすることしかない．また今後の高速化には，フォンノイマンボトルネックだけではなく，さまざまなボトルネックがからむ．

世の中すべてRISC

現在，最高速のRISCは，1CPUの場合，ほぼ現在最高速のメインフレーム(大型コンピュータ，RISCの10～100倍の価格である）なみの速度を持っている．速度においてRISCに対抗できる可能性があるのはスーパーコンピュータのみである．そして，大型コンピュータのようなCISCと異なり，RISCの速度はまだまだ速くなることが期待できる．したがって，5～10年後には，世の中のすべてのコンピュータはRISCになるであろう．つまり，RISCの将来とは，そのままコンピュータの将来と考えてよい．

では，今RISC技術に何が起こりつつあり，将来にどのような展望があるのだろうか？

チップ容量の増大

現在のLSIのトランジスタ数は，DRAM容量の推移を見ても分かるように，2～3年で4倍に増加している．近い将来，100万トランジスタや1000万トランジスタのCPUチップが出現することになるのは当然である．コンピュータはこういった事態に，どのように対応していくのだろうか？

一見すると，チップ容量の増大は，より多くの回路が必要なCISCに有利なようにみえる．しかし，実はCISCはすでに複雑さの限界に達していて，これ以上トランジスタが増加しても，もはやほとんど性能向上の役にたたない．

RISCの場合は，チップ容量の増大により，現在複数のチップに分かれているCPU周辺の要素が1チップにまとめられる．チップ間の通信に比べてチップ内の通信は高速なので，性能向上が期待できる．また，TLBなどすでにCPUに組み込まれている要素の数は増える．そして現在外付けになっているユニットも，チップ内に組み込まれていくことになるだろう．

最終的には，キャッシュメモリがCPUチップに載ることになるだろうが，その場合，浮動小数点ユニットのように高速動作する必要のあるユニットも，CPUチップに載せなければならない．さもないと高速のチップ内キャッシュのご利益が受けられないからだ．このようにして現在のCPUの周辺は，近いうちにすべて1チップに載ることになると思われる．

この1チップへの機能追加の可能性としては，まず現在のミニスーパーコンピュータなみのベクトル化された浮動小数点演算装置が考えられる．さらに集積度が上がると，複数のCPUが載ることになるだろう．

複数命令同時実行

RISCに関する技術で現在最も熱い領域が，複数命令同時実行である．50ページで登場した，仕事にかかる時間の式についてもう一度考えてみると，仕事あたりの命令数にはもはやそんなに変化する余地はない．また，クロック速度もそうは速くならない場合，性能向上のためにはクロックあたりの命令実行数を増やすしかない．

ところが，現在のRISCはすでにほぼ1クロックで1命令実行できるようになっているので，これからは，1クロックで2命令，4命令を同時に実行することになる．このための技術として，VLIW (Very Long Instruction Word) とスーパースカラー (Super Scalar) がある．

VLIWは，図13のように同じ演算ユニットやレジスタファイルを多数並列に並べて，それぞれに別々の命令を供給する方式である．とはいえ，いわゆるマルチプロセッサとは異なり，PCは全体で1個で，各ユニットの間で直接データ転送する命令も用意される．

一方スーパースカラーは，図14のように，見かけはこれまでのRISCと変わらないが，連続した命令が別々のユニットを使う場合について，それらを同時に実行してしまうよう，うまく制御する方式である．

●VLIW

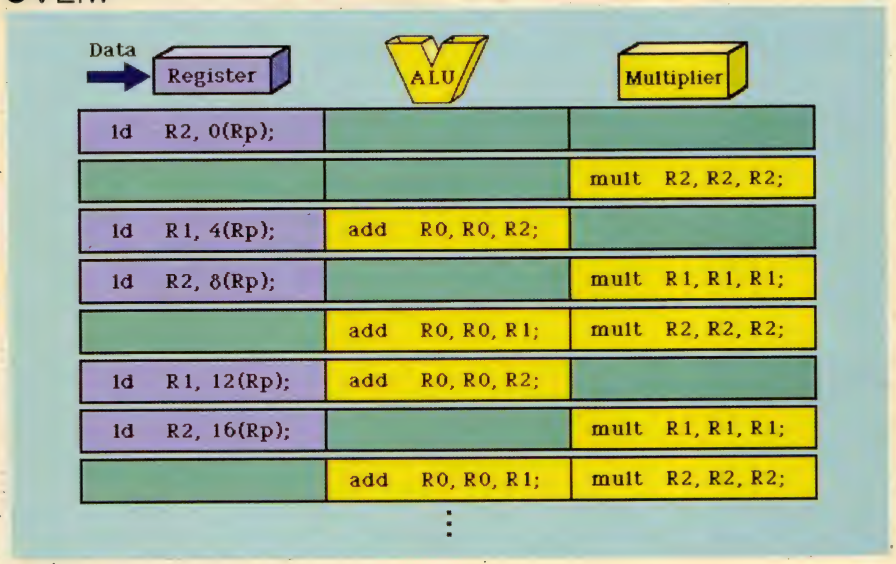


図13　通常のコンピュータの数倍から数十倍の長さを持つ命令は，分割してデコードされ，それぞれに対応する演算器への指示となる．結果として，1回命令を読むだけで，演算器の数だけの処理が同時にできることになる．すべての演算器に同じタイプの命令を与えることもできるが，この例では，メモリアクセスが同じタイミングで起きるのを避けるため，ロード命令の発行をずらすようにしている．

どちらの方式もプログラムのうち，もともと並列に実行可能であった部分を並列に実行するものであり，並列度もそれほどは上がらない．VLIWの場合，多数の演算ユニットとレジスタファイルを持ったところで，よほど都合のよい問題でないと，多くの演算ユニットが遊ぶことになったり，各ユニットの間で多くのデータ転送が必要になったりする．スーパースカラーの場合も，連続する命令が同じユニットを利用する場合や，前の命令の結果を後の命令が必要とする場合は，当然同時に実行できない．

したがって，どちらの方法を取るにしろ，複数命令同時発行による速度向上は通常2倍程度，せいぜい4倍が限界であろうといわれている．しかし，この速度向上もRISCだから可能なのである．CISCではもともとの命令が複雑なせいで，複数の命令が同時実行可能かどうか知ることは非常に複雑な問題になる．

VLIWはハードウェア的にはユニットを並べれば製作可能であるので，比較的やさしいほうだ．しかし演算器の構成は数からしてこれまでのものと変化するので，まったく新しいCPUを製作することになる．また，並列性の制御のかなりの部分はコンパイラが行なうことになるので，これまでのRISCのものよりもさらに高度なコンパイラの開発なども必要である．

いっぽうスーパースカラーの場合，2つの命令が同時実行できるかどうかの判定が複雑になって，クロック速度を低下させる危険がある．コンパイラが工夫して同時に発行できる命令の種類を制限し，うまくいかない場合は片方をNOPにするなどすれば，実行時の判定は簡単になる．しかし，この場合は従来のコンパイラで作成したプログラムが動作しなくなる．

マルチプロセッサ

複数のコンピュータを接続し，同時に動作させるマルチプロセッサ技術は，昔から処理の高速化には有効な手法であった．ただ，多くの応用では，コンピュータの台数を10倍に増やしても，すべてのコンピュータが同時に意味のある仕事をできるわけではなく，速度の増加は10倍以下である．ただ，2～4台程度のマルチプロセッサはさほど能率を落とさずに実現可能なので，現在最高速のRISCの速度をさらに高めるために利用されていくだろう．

マルチプロセッサには，コンピュータ同士の結合の強さによりいろいろな方式があり，結合が強いほど仕事の振り分けがやりやすく，効率的な動作が可能である．しかし，メインメモリの共有のような高速で強い結合は，10台程度が限界である．

これからの方向としては，より緩い結合ながら数百から数万台のコンピュータを柔軟なネットワークで結んで並行に動作させる，MPP(Massively Parallel Processor)が注目されており，大規模な配列計算のような規則的な計算で現在のスーパーコンピュータを置き換えていくばかりでなく，データベース処理のようなより複雑な処理もこなしていくとみられている．

I／Oの速度

これまでも，そして将来も半導体の速度はどんどん速くなっているが，I／Oの速度はそれほどには伸びていない．しかし，CPUが高速になれば，たとえば仮想記憶も高速でなければならず，これからは，高速のI／Oを行なう技術がより重要になってくる．

まず，ディスクの速度に関しては，メモリの場合のインターリーブと同様の，ストライピングという技術により向上させられる．ほかに，ディスクとCPUの間に，一時的にデータを蓄える半導体メモリを置くことも考えられる．突然のシステムクラッシュや停電が起きてもデータが失われたりしないように，半導体メモリの内容は，電池などでバックアップしてやる．

CPUの速度にネットワークの速度が追い付きそうにないことも，別の問題である．これから主流になるとみられる，光ファイバを利用した100MbpsのFDDIでも，現在最高速のRISCのデータ需要に追いつく程度でしかない．新しいネットワーク技術の規格が早急に登場することがのぞまれている．

●スーパースカラー

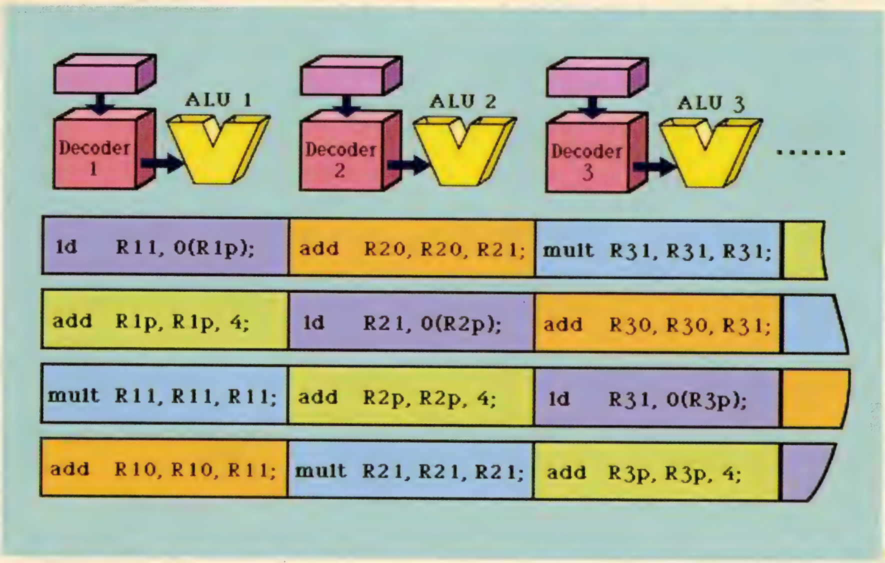


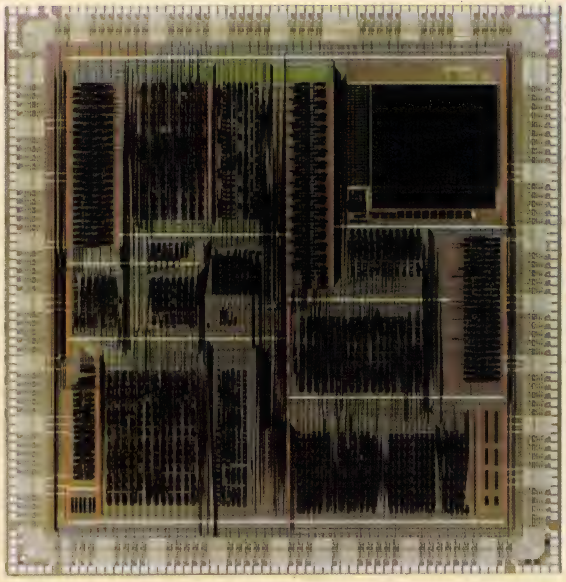
図14 命令のデコードは一括して行なわれ，それぞれの処理を行なうハードウェアに送られる．ハードウェア同士の独立性はVLIWの場合より低いので，並行して動作できるハードウェアの組み合わせは限られる．というより，特定の処理の組み合わせのみ並行して処理できる場合もスーパースカラーという．この例では，ループに登場するロード命令と加算命令をループアンローリングしてまとめたうえで，2命令分ずらして実行している．

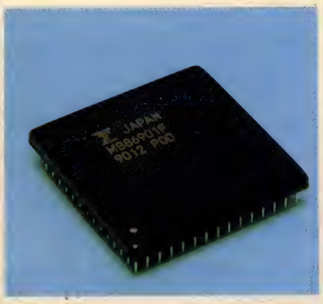
2 RISCの実現

現実のRISCは，ここまでに紹介してきた基本的な特徴のほかに，それぞれのコンピュータに特徴的な機構を持っている．現在一般的なRISC 4種についてその特徴と実現のされ方の研究，さらに性能の分析を行なう．この分析をもとに，今後のRISCのあるべき姿，すなわちコンピュータのあるべき姿について考えることができるはずである．

SPARC

かなり早い時期からSunワークステーションに使用されている一般的なRISC．レジスタウィンドウを採用しているのが，最大の特徴である．





Fujitsu MicrosystemsのMB86901．SPARCの整数演算ユニットである．規則的に並んだレジスタが，マスク上で大きな面積を占めているのが分かる．キャッシュ，MMUなどはすべて外付け．4段パイプラインを持ち，クロック周波数25MHz．1.5μmCMOSスタンダードセル技術により製作されている．消費電力は，1.4W．179ピンのパッケージに収められている．サンプル価格は4万8000円．

背景

SPARCアーキテクチャは，カリフォルニア大学バークレー校でのRISCに関する研究成果をもとに，Sun Microsystems社によって開発された．

Sun Microsystems社といえば，1982年に創設されたワークステーションの老舗であり，モトローラ社の68000系のCPUを用いたSun2／Sun3ワークステーションシリーズにより，業界のトップメーカーとなった．Sunマイクロシステムズ社が1987年に発売したSun4で，初めて採用されたアーキテクチャがSPARCである．

発表当初のSun4は，16.67MHｚのクロックで公称10MIPSの速度を誇り，実際にもSun3シリーズの2倍以上の速度は出せた．現在も，SPARCはSPARCstationシリーズなどに利用され，公称20MIPSとなっている．SPARCアーキテクチャのチップは，現在，Fujitsu Microelectronics，Cypress Semiconductorなどが，整数演算チップとメモリ管理ユニットなどを発売しており，汎用の浮動小数点チップと組み合わせて使用されている．TLB，キャッシュなどは，チップセットには含まれておらず，外部回路で構成する．

レジスタウィンドウ

SPARCのアーキテクチャの最大の特徴は，レジスタウィンドウである．

MB86901では，120個の32bit汎用レジスタを持ち，それだけでチップの面積の1/4を占めている．うち8個はglobalレジスタであり，常にアクセス可能だが，残り112個は，8個ずつのin，local，outレジスタからなる24個のレジスタのグループにまとめて使用される．このレジスタのグループをレジスタウィンドウ（窓）と呼ぶ．これらのグループはルーチンの呼び出しごとに専用に用意され，それぞれのルーチンの中からは32個の汎用レジスタがアクセス可能である．

localレジスタは，ルーチンの局所変数を保持し，inレジスタはそのルーチン自身への引数が入っている．サブルーチン呼び出しを行なう場合，渡すべき引き数は

タイプ	種類	フォーマット		注
Load／Store	22	ld	[%fp+0x20]，%o0	ロードには，データの長さにより2～3クロック，ストアにも3～4クロックかかる．別のアドレス空間を対象にした命令もある．
Arithmetic	31	add	%g1，%lo(CONST)，%g1	通常の加減算，シフトなどのほか，Lispなどのためにタグ付きの加減算も用意されていて，それも1サイクルで終わる．乗除算はない．
Jump／Branch	5	be	LABEL	整数演算の結果をもとにしたブランチや，ルーチン呼び出しのためのcall命令などが用意されている．
Special	8	rdy	%g2	WIM（Window Invalid Mask），TBR（Trap Base），Y（multiply-step）などのレジスタに対する読み書きができる．
Floating	(21)	fadd	%f8，%f6，%f8	浮動小数点命令は，SPARCチップに実装されておらず別のプロセッサで実行されるが，命令フォーマットは定義されている．
総計	66+21			%で始まるシンボルは，レジスタの名前．英字がレジスタウィンドウのグループ分けに対応する．

SPARCの命令セット．命令セットは比較的小さい．命令の実行クロック数はMB86901のもの．命令フォーマット的には命令部分を3種類の長さに分け，call命令に30ビットのオペランドが使えるようにしたのが特徴である．

outレジスタにおく．またこのとき，レジスタの切り替えが行なわれ，これまであったoutレジスタが新たにinレジスタとなり，localとoutレジスタには，これまで使われていなかった部分が割り当てられる．この方式には関数呼び出しがレジスタウィンドウを指すポインタの切り替えだけで行なえるという利点がある．

この例では，7個の関数について，レジスタウィンドウが用意できることになる．関数呼び出しが起きたときに空いているレジスタがなくなってしまっていた場合には，割り込みが起き，レジスタをメモリに転送して空きを作る．通常のプログラムでは2～3レベルの深さの関数呼び出しがほとんどであるので，このメモリへの転送は，性能上問題になるほど頻繁には起きない．

レジスタウィンドウは，一見便利なメカニズムであるが，RISCという観点から見ると疑問点も多い．関数呼び出しの際に，必ず16個のレジスタが新たに用意されるわけだが，この数は最適なものとは限らず，本来コンパイラによって解析された値が利用されることが望ましい．レジスタウィンドウのための専用ハードウェアを用意するよりは，高度な最適化コンパイラによるグローバルレジスタ割り当てのほうがRISC的なアプローチといえよう．また，レジスタウィンドウのために全体で120ものレジスタがあるわけだが，レジスタの数は多いほうがいいわけではない．数が増えると，レジスタの番号から，レジスタに入っているデータを取ってくるためのデコード回路の段数が増え，レジスタをアドレスするための時間が無視できなくなるからである．一度に利用するレジスタが32個で済むなら，デコード回路も32個アドレスできる分だけで済ませたいものである．

乗除算

SPARCのアーキテクチャで，もう1つ問題になるのは，整数乗算／除算命令の欠如である．SPARCでは，整数の乗算を直接行なう命令は存在せず，その代わりに乗算ステップ命令というものがある．乗算ステップ命令では，1命令では1bitと32bitの乗算しか行なえず，32bitと32bitの乗算は，プログラムでループを32回組んで行なう．除算にいたっては，専用命令はまったく存在しないため，1bitあたり9クロックの時間が必要である．

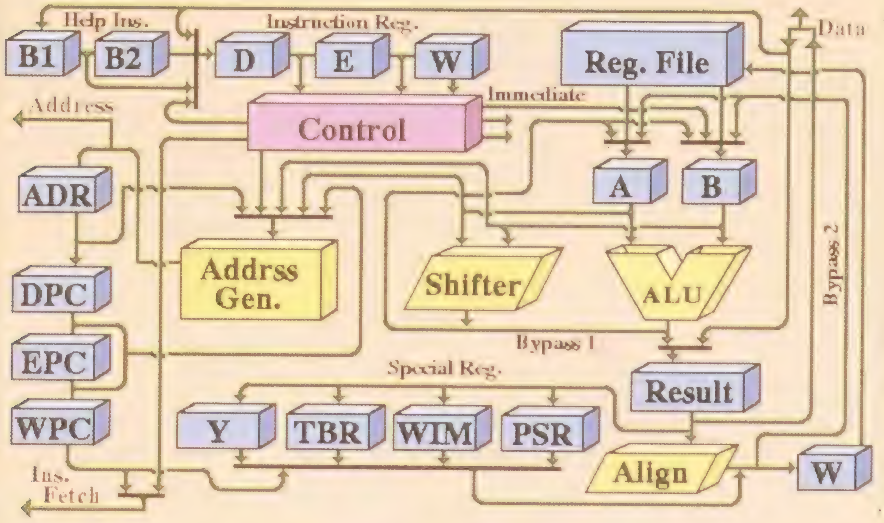
このようなアーキテクチャでも，一般のプログラムには整数の乗除算はあまり出現しないため，速度はほとんど低下することがない．最適化コンパイラを利用すれば，ループ中の配列の参照などはストレングスリダクションにより，定数の加算に還元できる．残りの乗算もほとんどが小さな定数との乗算であり，シフトと加算に還元可能である．たとえば，ある数に9を掛けるには，$9 = 8 + 1 = 2^3 + 1$であるから，その数を3bit左シフト（8倍することに相当）したものにもとの数を足せばよい．

そこで，RISCの設計に際して通常のベンチマークを行なうと，整数の乗除算は不要という結論がでがちである．しかし，世の中には整数演算を集中的に行なう応用も無視できない程度存在し，そのような応用のSPARCでの実行効率は著しく低下することになる．ただし将来のSPARCでは，専用乗除算命令が用意される可能性もある．

将来

現在SPARCアーキテクチャに従ったチップとしては，BIT社による50MHz程度のECLチップが発表されている．

さらに野心的な試みとして，GaAs(ガリウムヒ素)技術を利用したクロック250MHzのチップがプリズマコンピュータ社により開発されていたが，同社は倒産してしまった．この失敗の原因は主に技術的困難であり，現在のGaAsチップの集積度ではCPU製作に必要な容量がえられないことと，CPUの周辺として利用可能なチップがGaAsのCPUに比較して十分高速ではないこと，配線の遅延が無視できなくなること(250MHzでは光ですら1クロックに1.2mしか進めない)などの問題が大きい．



4段のパイプラインのうち命令フェッチ以外のステージに対応して，D，E，Wの3つの命令レジスタと，DPC，EPC，WPCの3つのプログラムカウンタが存在し，割り込みの場合の状態の回復を容易にしている．直前の命令の演算結果がすぐ必要な場合，レジスタファイルへの書き込みを待たず，Bypass 1，2を使う．

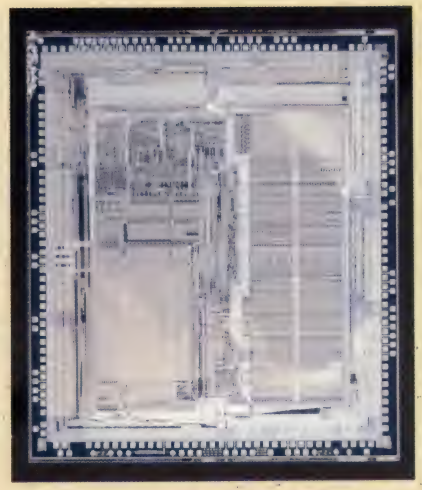
性能 ●SPARC，25MHz ●SPARCstation330

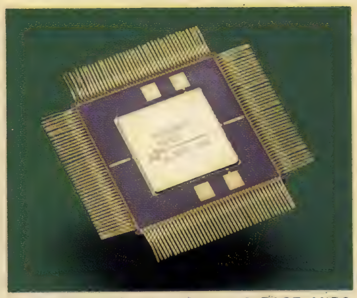
Integer	12.2
Floating Point	11.5
SPECmark	11.8

SPECmarkは，SPECという団体によるベンチマークで，VAX11/780との性能比を表わす．Integerの数値は，おおむね1MIPSに相当する．

R3000

R3000はMIPSアーキテクチャに基づいて作られたRISCであり，コンパイラによるパイプラインの効率維持を重視している．





Performance Semiconductor の PACE MIPS R3000A. CPUとしては，非常に規則性が高いマスクである．0.8μmCMOS技術により作られており，クロック周波数33MHzで消費電力は最大3.57W．写真のフラットパッケージは172リード．ほかに175ピンのPGAもある．そちらのサンプル価格は8万円．

背景

MIPSのアーキテクチャは，1981年からのStanford大学におけるRISCに関する研究成果をもとにして作られている．MIPS社はこの研究の関係者を中心に1984年に創設された会社で，当初よりMIPSアーキテクチャの開発を中心業務としている．

面白いことにMIPS社自体ではLSIのマスクパターンの設計のみを行ない，チップ自体はすべて他社にライセンス生産させている．しかし，その他社のチップをもとに，ワークステーションの制作や，そのOSやコンパイラの開発も行なっている．RISCでのソフトウェアや全体のバランスの重要性を考えれば，このように自社で完全なコンピュータを制作し，その結果を次期システムに反映するのは当然のことであろう．

チップセット

MIPS社のCPUは1986年に発表されたCMOS，動作クロック8MHzから16MHzのR2000シリーズと1988年のR3000シリーズ（CMOS，20MHz～40MHz），1989年のR6000シリーズ（ECL，66MHz）の3種類である．R2000とR3000は動作クロックを除けばほとんど同じものだが，R6000にはいくつかの機能が追加されている．

以後は現在主に使われているワークステーション用のR3000シリーズを中心に解説する．R3000シリーズは，3チップで構成され，R3000チップは，整数演算や制御を行なういわゆるCPU部分と，システムコントロールプロセッサと呼ばれるコプロセッサを含む．R3010チップは浮動小数点コプロセッサであり，R3020チップがライトバッファと呼ばれるものである．

R3000のCPUには，32個の32bit汎用レジスタと，2つの乗除算用のレジスタと，プログラムカウンタがある．汎用レジスタ0の内容は常に0である．一般的なコプロセッサインターフェイスが公開されており，チップ内部のシステムコントロールプロセッサのほか3個のコプロセッサが付加できる．

システムコントロールプロセッサでは，割り込みの処理や仮想記憶の管理を行なう．また64個のTLBも内蔵する．キャッシュ自体はチップセットの外部にあるが，キャッシュコントローラは，R3000チップに内蔵されており，4～64Kbytesのデータ／命令キャッシュが制御できる．

R3020ライトバッファは，メインメモリへの書き込みのためのバッファである．メインメモリからの読み出しでは，CPUは完了を待ってデータを受け取らなければならないが，書き込みでは書く内容とアドレスさえ確定すれば，そのあとはCPU

●Integer Unit

タイプ	種類	フォーマット	注
Load／Store	12	LB　rd, offset16(base)	ロードストアは，ベースレジスタから16bitのオフセットを指定するモードしかない．32bitを1命令でロードすることはできない．
ALU immediate	8	ADDI　rd, rs, immediate16	オペランド中の16bit長データとレジスタRsとの間で演算し結果はRtにセットされる．R3000では演算の方向は右から左．
Register type	10	ADD　rd, rs1, rs2	レジスタを3個指定してレジスタ間の演算をする命令もある．大小の比較をする命令もこのタイプ．
Shift	6	ALL　rd, rs, shamt5	シフトする量をレジスタで指定する命令と，ダイレクトにオペランドで指定する命令が用意されている．
Multiply	8	MULT　rs, rs	乗除算は他の処理と並行して進められ，終了後，結果はHI，LOという専用のレジスタの組にセットされる．
Jump／Branch	12	J　target	ジャンプする先のアドレスは，26bitで指定されたものを2bitシフトして，さらに現在のPCの上位4bitと加えて求める．
misc.	2	SYSCALL	ソフトウェアによる割り込みのための命令．システムコール用と，ブレーク用の2種類がある．
総計	74		ここまでに挙げた命令に加えて，コプロセッサとのインターフェイスの命令と，TLBの操作に関わる命令がある．

●Floating Point Unit

タイプ	種類	フォーマット	注
Load／Store	6	LWC1　ft, offset(base)	浮動小数点レジスタとメモリとのデータのやり取りは，レジスタを2つに分割し，32bit単位で行なわれる．
Add／Sub	5	ADD　fd, fs, ft	加減算はレジスタ同士で行なわれ，2サイクルかかる．符号反転や絶対値などの命令も用意されている．
Mult／Div	2	MUL　fd, fs, ft	乗除算は，4～19サイクルかかるが，整数演算の場合と同様に，ほかの演算を並行して進めることができる．
総計	14		このほか，浮動小数点レジスタの値を比較して，コンディションコードを設定する命令も用意されている．

R3000の命令セット．上に挙げた機能のほか，ワード境界に合っていないワードを2命令でロードする機能がある．またMIPS IIでは，浮動小数点データを1命令でロード／ストアできるようになっている．

が介入する必要がない．そこで，ライトバッファを用意することにより，CPUからは1クロックでメモリへの書き込みが終わるように見える．

ソフトウェア

R3000の特徴は，とにかく速度を落さずにできることは何でもソフトウェアで行ない，なるべく専用のハードウェアを用意しないことである．レジスタウィンドウを用意する代わりに，大域的なレジスタの最適化によってレジスタの退避を省こうとするのもその一例である．また一般のコンピュータではメモリアクセスに際してTLBに対応するアドレスが存在しない場合，ハードウェアがページ変換テーブルを自動的にひくことになる．ところがR3000のハードウェアでは割り込みが発生するだけであとの処理はソフトウェアに任される．TLBがミスした場合には，データが読めるまでに，数十クロックかかるので，ソフトウェアで処理しても性能への影響は少ない．

なお，この割り込みの処理のためにOSへのコンテキストスイッチを普通に行なうと，さらにメモリアクセスが起き，TLB変換ミスが起きる可能性がある．これを防ぐため，TLBミスの処理専門に2つの汎用レジスタを割り当て，このレジスタだけで，処理を済ませている．ユーザーのプログラムでは，この2つのレジスタは，利用できない．

また，ハードウェアで識別される割り込みの種類には，このTLBのミスとリセットしかなく，ほかの割り込みの原因の解析はすべてソフトウェアで行なう．

またMIPS(Microprocessor without Interlocking Pipeline Stages)の名のとおり，完全なインターロック機構は持たない．一般にメモリアクセスを行なう命令では，たとえキャッシュ内にデータが存在しても，直後の命令に間に合うようにデータを用意することはできない．そこで，直後の命令が前の命令でのメモリアクセスの結果を利用する場合，実行時に検知し直後の命令の実行をデータが利用可能になるまで遅らせる必要がある．これをインターロックという．

しかしR3000では，この処理を行なわない．インターロックが必要であるかどうかはコンパイル時にすべて検出できるので，ハードウェアで実行を遅らせる代わりに，コンパイラが余分なNOP命令を配置する．なおコンパイラが検知している以上，直前でロードしたデータに依存しない命令を移動してくることができれば有効な命令を実行させられる．

ただし，浮動小数点演算や，整数乗除算など，1サイクルでは実行が終わらない命令についてのインターロックはハードウェアが自動的に行なう．これは，データの内容や浮動小数点プロセッサなどの性能によって待つ時間が大きく変動するため，コンパイル時には無駄の多い解析しかできないからである．

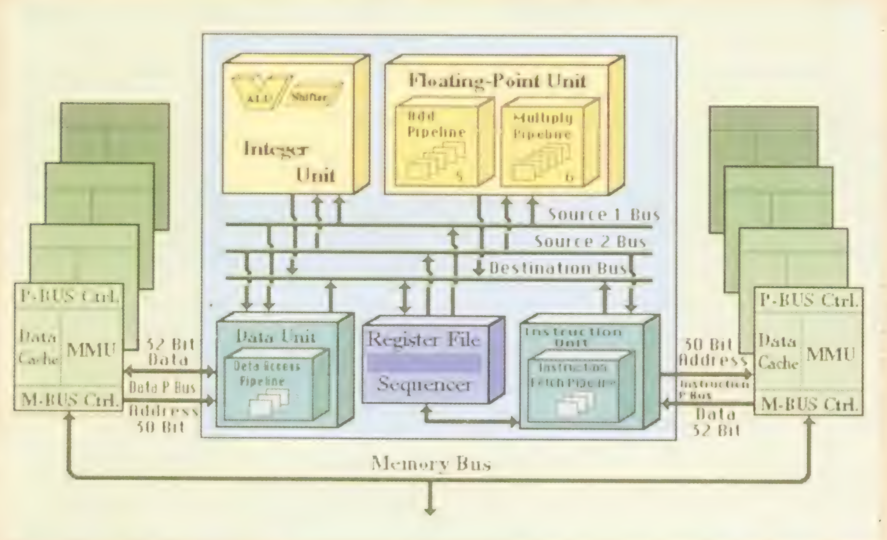
R3000のもう1つの特徴は，1クロック中にデータキャッシュと命令キャッシュをそれぞれ1度ずつアクセスできることである．データアクセスが起きても，命令の取り出し速度は影響を受けない．

将来

ECLのR6000は，公称55MIPSである．一口に55MIPSというが，この速度は，1CPUの性能では，通常の大型コンピュータをしのぎ，不規則性の大きい計算ではスーパーコンピュータに匹敵する．また，R4000と呼ばれる，1チップに浮動小数点ユニットやキャッシュも内蔵したCMOSの高速版もうわさされている．また，R6000のほうもクロックの高速化が予定されている．

R6000やR4000では，命令体系がMIPS IIアーキテクチャと呼ばれる拡張されたものになった．これにより，ジャンプのディレイスロットの効率的な利用などが行なえる．

また，メインメモリとCPUの速度差が大きくなるため，キャッシュは2レベルになる．R6000では1次キャッシュとして，命令用に64Kbytes，データ用に16Kbytesの容量を持つが，命令／データ共用の2次キャッシュは，なんと2Mbytesバイトまで持つことができる．



チップ内部は，整数演算を行なうCPU部分と，アドレス変換と例外処理を受け持つコプロセッサCPOに分かれる．割り込みの解析はもちろん，割り込み復帰の際の複雑な状態回復も，Exception Ctrl. Registerの内容から，ソフトウェアで行なう．乗除算に関して，32bitの専用のレジスタが2つある．

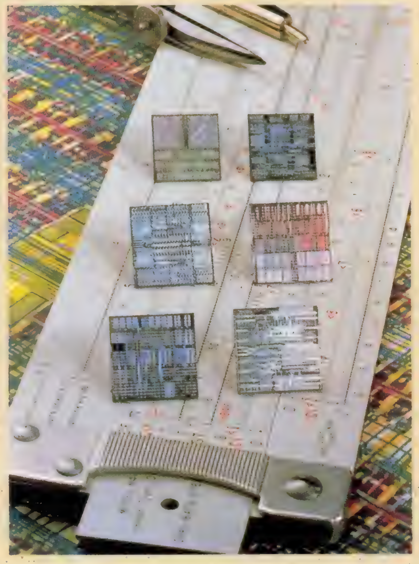
性能 ●R3000，25MHz ●RS3230	
Integer	19.4
Floating Point	16.8
SPECmark	17.8

特に苦手な分野というものはなく，各テストに対して平均的な数値を出している．公称MIPS値にも近い．

RS/6000

RS/6000はIBMによって商品化された２つ目のRISCで，浮動小数点演算を重視している．１クロックに２つ以上の命令を実行できる．

左の写真はPOWERserver．左上に並んでいるのがRS/6000のチップセットである．右上がメモリボード．下のチップ写真は，左上がDCU，右上がSCU，以下I／Oユニット，ICU，FXU，FPUとなる．すべて，1μm CMOS技術によって作られている．DCU4個の構成で，ロジックが204万トランジスタ，メモリが486万トランジスタとなる．



背景

IBMはいわずと知れた世界最大のコンピュータメーカーであり，その商品は大型汎用コンピュータからパーソナルコンピュータに至るまで多岐にわたる．しかしワークステーションに関しては，1986年にRT/PCを発売したものの，あまり反響はなかった．そのIBMが今年ワークステーション業界に送り込んだのが，RS/6000である．

そもそも，RISCというアイデア自体，IBMで1974年に始められた高速電話交換用のコンピュータの研究から生まれたものである．801と呼ばれるそのコンピュータは，別々の命令／データキャッシュ，遅延分岐，高度なコンパイラなど，現在のRISCの基本的な特徴を，当時すでに備えていた．

この研究は，カリフォルニア大学バークレー校やスタンフォード大学に受け継がれて，MIPSやSPARCへと発展したわけだが，IBM内部でもRISCの研究開発は続けられていた．

チップセット

RS/6000は，クロック速度20～30MHzで動作する命令キャッシュユニット(ICU)，整数ユニット(FXU)，浮動小数点ユニット(FPU)，データキャッシュユニット(DCU)，記憶制御ユニット(SCU)，I／Oユニットと多くの専用チップからなる．

ICUは，8 Kbytesの命令キャッシュを内蔵し，命令の取り出しやデコード，ジャンプ命令の実行をつかさどる．プログラムカウンタもICU内部に保存されている．単純なカウンタによるループ命令は，内蔵されたカウンタによって実行され，ブランチによる遅延は起きない．内蔵された4個のデコーダは並列に動作し，命令キャッシュから命令を先読みする．さらにブランチ先アドレスの生成を行なう．また，必要に応じてFXUやFPUの制御を行なう．

FXUは，32個の32bitレジスタを持ち，整数の加減乗除，シフトなどを行なう．また，128個のTLBも内蔵し，アドレスの計算やデータキャッシュの制御も行なう．

FPUは，32個の64bitレジスタを持ち，

●Fixed-Point Processor (FXU)

タイプ	種類	フォーマット	注
Arithmetic	24	AI　r3=r3,8	加減乗除など通常の演算はそろっており，それぞれに拡張精度も設定できる．また，最大，最小，絶対値などの命令も用意されている．
bit-operation	42	RN　r3=r3,9,0x1ff	通常のbit演算もすべてそろっている．複数のレジスタにわたるbitストリングについてシフトなどを行なうこともできる．
branch	13	CL　cr1=r3,100	ICUとの間でデータをやり取りするための命令．コンディションコードを設定するcompare，割り込みを起こすtrapなどがある．
Load／Store	55	L　r3=.symbol(r2.0)	ロード／ストアには各種のアドレスモードがありインクリメント／デクリメントもできる．CやPascalの文字列のための命令もある．

●Floating-Point Processor (FPU)

タイプ	種類	フォーマット	注
Arithmetic	13	FMA　fp1=fp1,fp2,fp0	FMAは，レジスタ２つの積にもう１つのレジスタを加算し，さらに別のレジスタに書き込める．
FPSCR	6		FPSCR(Floating-Point Status and Control Register)をストアしたりロードしたりする命令が用意されている．
Compare	2	CFL　cr1=fp1, fp2	浮動小数点レジスタを比較してコンディションコードをICUに設定するための命令．

●Branch Processor (ICU)

タイプ	種類	フォーマット	注
jump／branch	15	BT　LABEL, cr0, 0x1/lt	ブランチ命令は，ICUで処理される．コンディションレジスタ内のbitについて論理操作をする命令もある．
SVC	1	SVC	ソフトウェア割り込みを起こす命令．コントロールレジスタや，PCなどの退避は，すべてICU内部で行なわれる．
総計	184		平均的なCISCマイクロプロセッサを上回る，極めて多くの命令が存在する．しかし，命令構成はCISCとは異なる．

並列に動作するユニットが多いので，処理の全体を把握するのは難しい．命令も多く用意されているので，コンパイラの作成は容易ではあるまい．しかし文字列操作などの一見過剰に思える命令も，目的があるようだ．

浮動小数点演算を行なうが，特に，浮動小数点乗算と浮動小数点加算を複合した演算fmaを1クロックサイクルに1つずつ実行できる．またレジスタロードのインターロックを，ダイナミックな名前の付け換えによって回避している．

DCUは，16Kbytesのデータ用キャッシュであり，システム構成に応じて，2個または4個使われる．DCUとメインメモリとの間では，データは128bytesという大きな単位で転送されるが，これだけの量が転送されるまでFXUやFPAが待つわけではない．読み出しの際のキャッシュミスがたとえば128bytesのうちの20～23bytes目に対して起きた場合，まず，その20～23bytesの部分がメモリから転送され，FXUやFPAなどに渡され，残りの124bytesは，その後で読まれる．

SCUは，CPUとメインメモリ，I／Oとの通信の制御を受け持ち，CPUの指示により適切なメモリ制御の信号を発生する．また，定期的にメインメモリの内容を検査し，修正可能なエラーがあれば，これを修正したり，メモリICの故障を検出したりしてくれる．

I／Oユニットは，マイクロチャネルと呼ばれるIBMの標準的なI／Oバスへのインターフェイスのほか，2つの光ファイバインターフェイスによりRS／6000同士を400Mbpsの速度で直結する機構を提供している．

スーパースカラー

RS/6000の最大の特徴は，スーパースカラー技術が採用されていることである．ICUとキャッシュは128bitのバスで結合されており，1クロックの間に32bitの命令が4つ同時に読み込め，別々のユニットを操作する命令は，すべて同時に実行できる．つまり，FXU，FPA，ICUの各ユニットに対する命令は，相互の干渉がないので1クロックで同時に実行されるが，FXUを利用する命令2つが同時に実行されることはない．

RS/6000は，特に数値計算での応用を重視した命令体系となっている．数値計算に現われる最も典型的なループは，配列の乗算などであり，浮動小数点乗算と浮動小数点加算の複合命令(FPAにて1クロックで実行可能)と，ポインタの更新(FXUで実行)，ループの終了判定のための条件ジャンプ(2命令になるがICUで1クロックで実行可能)で構成できるため，1クロックあたり1回ループが回れることになる．クロックが30MHzの場合，理論最大演算速度は60MFLOPSであるが，実際にもLINPACKと呼ばれる浮動小数点演算性能のベンチマークテストで13MFLOPSが測定されている．

ただ，この速度を達成するためには，かなり条件が多く，整数演算や多少とも毛色のかわった浮動小数点演算への応用では，これほどの速度は出ない．また，浮動小数点乗算と浮動小数点加算の複合にも無理があり，通常の乗算と加算の組み合わせに比較して精度的に多少劣ることになる．ただしRS/6000は途中の結果を通常の3倍の156bitの精度で持ち，最終的な精度を保っている．

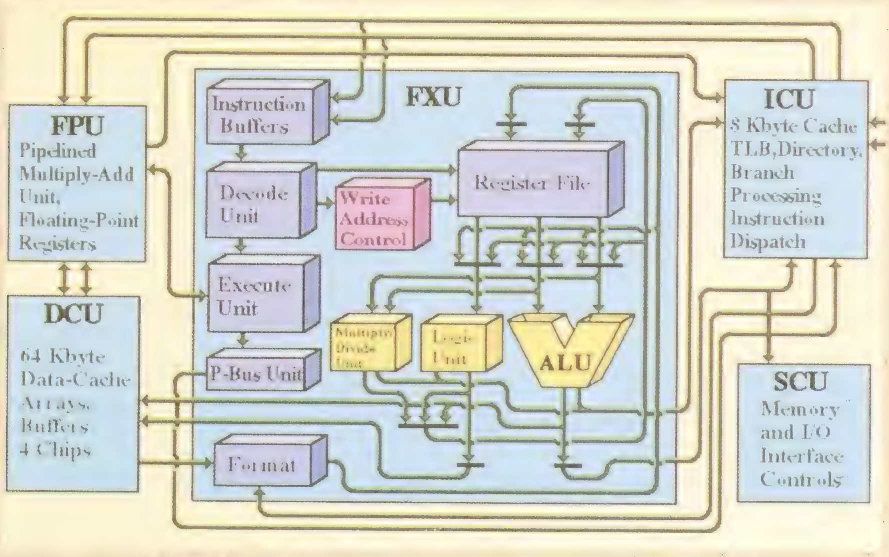
評価

全体の命令数は184個と非常に多く，これだけみるとRISCの名に値しない．しかし，命令が増えた原因のある程度は，CPUを構成するプロセッサが分離されたことによる．またパイプラインの効率的な利用による高速な命令の実行は，RISCの特徴である．

RS/6000のSPECmarkは非常に高速であるが，これは，浮動小数点演算を利用する場合の速度が突出していることが大きい．FXUでの多重命令の実行は行なえないためスーパースカラーとしての能力は，限定されているが，整数計算でも同一クロックのR3000と同程度の能力は持つ．

将来

これまで，IBMは370アーキテクチャと呼ばれる製品群で汎用大型コンピュータ業界に君臨してきた．しかし，汎用大型コンピュータとワークステーションの将来性を比較するとき，IBMにとってRS/6000の成否は非常に重要になるだろう．



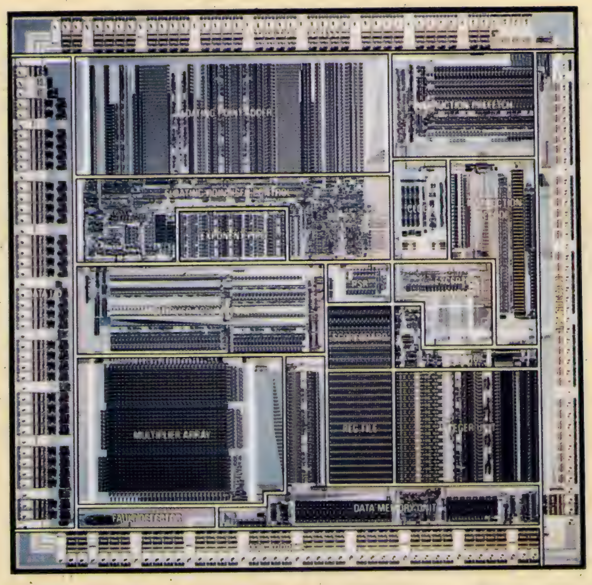
ICUはFXU，FPUに同時に命令を送れる．FXUでは，ICUより送られてくる命令をInstruction Buffersに蓄積して，順次実行する．FPUとの間で，実行する命令が離れ過ぎないように，Execute Unitを通じて同期をとる．FPUとDCUの間のデータ線は64本あり，1クロックに1浮動小数点数のデータ転送が行なえる．

性能	●RS/6000, 25MHz ●prototype
Integer	——
Floating Point	——
SPECmark	28.9

25MHzのPOWERserver530で，Integerで20，Floatingで37という結果もある．ただしFloatingはテストにより大幅に結果が異なる．

MC88000

88000は，これまでマイクロプロセッサの発展の中心となってきたモトローラによるRISCであり，専用のCMMUを持っている．



モトローラのMC88100．CMOSのチップ上では乗算器が大きな割合を占めている．全体に浮動小数点関係の要素が占める割合が大きい．命令プリフェッチにも意外に大きなスペースが取られている．このチップが180ピンPGAのパッケージに収められており，サンプル価格は6万3000円．ちなみにCMMUのMC88200のほうも，同じパッケージに収まっており，7万7000円．

背景

モトローラは，インテルとならびマイクロプロセッサの最大手メーカーである．従来，典型的なCISCである68000シリーズのCPUを発売してきたが，世の中のRISC化の勢いには逆らえず，1988年に88000の発表となった．モトローラは，その後も68000シリーズの拡張を続けているが，今年発表された最新の68040は，88000より低速である．

チップセット

88000ファミリーは，88100CPUと88200 CMMU（Cache／Memory Management Unit）から構成される．キャッシュを専用チップで用意したことにより，システムの構成は容易になっている．

88100CPUは，32個の32bit汎用レジスタを持ち，整数の加減乗除のみならず，浮動小数点演算ユニットまで内蔵している．浮動小数点用レジスタは特に持たず，汎用レジスタがそのまま浮動小数点の演算の対象となる．

また，この浮動小数点演算ユニットのほかに，同じインターフェイスで6台までのSFU（Special-Function Unit）が接続できる．このようなインターフェイスの利用法としては，疎結合マルチプロセッサを実現するための，高速の通信リンク，あるいはベクトル演算器などが考えられる．88000はRISCとしては後発であり，独自の特徴を出そうという努力がうかがえる．

整数演算としてはbitフィールド操作の命令群が充実しており，ウィンドウシステムのサーバとしては多少の速度向上が期待される．アドレスモードも比較的豊富であり，2つのレジスタを加算してアドレスとしたり，2つのレジスタの片方を2～8倍にして足すようなモードがある．しかし，このような機能がRISCとしての全体の性能向上に，どれだけ役立つかは疑問である．

88200CMMUは，1チップに56個のTLBと16Kbytesのキャッシュを載せ，データキャッシュとしても命令キャッシュとしても使用可能である．キャッシュ方式として4多重セット連想方式と呼ばれるものが採用されている．この方式はRS/6000でも採用されているが，キャッシュの入れ替えが起きる可能性が減少し，ダイレクトマップ方式と呼ばれる一般のキャッシュ方式と比べて，倍以上のキャッシュ容量があるのと同等の性能が出る．

また，CMMUはデータ，命令のそれぞれについて，任意の個数並列に配置できる．この構成では単一のキャッシュの容量が増大するのではなく，アドレスをページ単位で切り分け，CMMUの個数おきに同じCMMUを割り付ける．これによりCMMUへのアクセスが平均化され，CMMUを追加することによって性能が向上することが期待できる．

CPUからは命令用バスとデータ用バスが別々に出ており，それぞれ別のCMMUに接続される．この命令とデータのバスを分離する方式は，ハーバードアーキテクチャと呼ばれ，データアクセスと命令フェッチが衝突を起こすことなく同時に行なえる．

また，各チップの内部では，比較的低速の外部クロックをもとにより高い内部クロックを発生して，外部クロック以上

タイプ	種類	フォーマット	注
Load／Store	7	ld　rd，rs，immediate16	2つのレジスタのアドレスを加算したものを使う命令など，CISCのアドレスモードに相当するものが用意されている．
Logical	4	AND　rd，rs，immediate16	bit単位の論理演算はマスク，AND，OR，XORと，必要最小限の演算だけが提供されている．
Arithmetic	8	ADD　rd，rs，immediate16	整数の数値演算は乗除算を含めてそろっている．ただし乗除算には浮動小数点ユニットが使用される．
bit-field	8	CLR　rd，rs，width5(offset5)	通常のシフトなどの演算の代わりに，より強力な，指定した範囲のbitの移動，取り出しといった演算が用意されている．
Jump／Branch	12	J　rs	アドレス指定は，ジャンプについてはレジスタで，ブランチについては26bit長のオペランドで直接指定する．
Floating	12	FADD　rd，rs1，rs2	浮動小数点命令は加減乗除のほか，比較などができる．除算以外は，5～9クロックで実行できる．
総計	51		例外的な命令はない．TRAPなどは，Jump／Branch命令のカテゴリーに含まれている．

命令の種類は，RISCとしても極めて少ない．そもそも命令の種類のエンコーディングに6bitしか取られていない．bitフィールド以外の命令は，すべてあって当然というものばかりである．

に細かなタイミングの制御を行なっているが，PLL (Phase Locked Loop) という技法により，異なるチップの間の内部クロックのタイミングも同期がとれるようになっている.

マルチプロセッサ

88000のアーキテクチャの最大の特徴は，メモリを共有した密結合マルチプロセッサに対応していることである．複数のCPUで同じメモリを共有したマルチプロセッサでは，片方のCPUで行なったメモリへのデータの書き込みが別のCPUからの読み出しにも反映していなければならない．1階層だけからなる単純なメモリ構成では，同じアドレスのデータは1つしか存在しないのでこのことは自動的に保証される.

しかし，CPUにキャッシュメモリを付加している場合には，メモリ中のデータのコピーがキャッシュ内にも存在するかもしれない．このような場合にキャッシュとメインメモリの内容に矛盾が生じないようきちんと管理する問題をキャッシュコヒーレンスの維持と呼ぶ．キャッシュコヒーレンスを維持する効率的な方法として知られているのが，スヌープキャッシュの技法である.

スヌープキュッシュ方式では，各キャッシュメモリは，メインメモリへのバスを常に監視している．そして，もし，あるキャッシュが保持しているデータのアドレスに，別のCPUからデータが書き込まれた場合，そのキャッシュは自分の持っているデータのコピーを無効になったものとして捨ててしまう.

このようにすれば，書き込みが起きたアドレスのデータのコピーは，最も最近にそこに書き込みを行なったCPUのキャッシュにしか存在しないため，キャッシュコヒーレンスは維持できることになる．一方，読み出しのみが行なわれるアドレスに対しては，複数のキャッシュがそれぞれコピーを持てることになる．また，この方式では他のCPUの監視はキャッシュメモリが行なうため，CPUにはコヒーレンス維持のための負荷はかからない.

88200CMMUは，このスヌープキャッシュのメカニズムが組み込まれており，88000では簡単にマルチプロセッサシステムを構成できる．マルチプロセッサシステムでは，同じ仕事をするCPUが相互の邪魔をしないように，一方が重要なデータを操作している間，他方がその完了を待つ必要があるが，そのための待ち合わせを行なうセマフォのメカニズムもCMMUに備えられている.

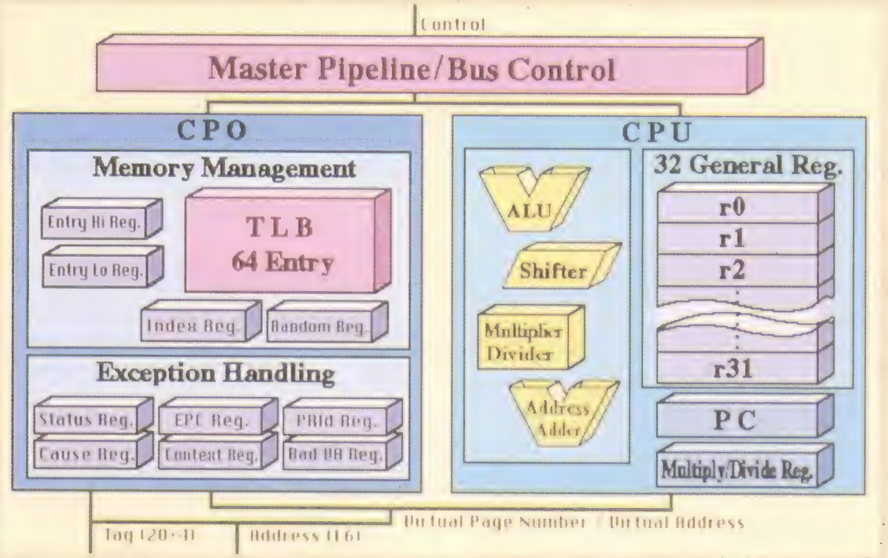
88000のもう1つの大きな特徴は，フォールトトレランス機能である．フォールトトレランス機能とは，故障や雑音による機能障害を検出し，回避するための技術である．故障の修理が事実上不可能な無人宇宙船や，故障が重大な経済的損失を生む銀行のオンラインシステム，人の生死にかかわる軍用／医療用機器などに利用されている.

88100，88200ともに，あるピンの設定により，チェッカーモードで動作させられる．チェッカーモードでは，外部からの信号の入力は通常どおり行なうが，出力の代わりに，外部の信号が自分が本来出力すべきである信号と同じであるかどうかのチェックのみ行ない，もし違いがあれば，エラー信号を出力する．そこで，2つ以上の88100や88200を並列に接続し，1台以外をチェッカーモードで動作させれば，チップの故障や雑音による誤動作を簡単にチェックできる.

チェッカーモードでは，誤動作の検出が行なえるだけである．しかし，このようなシステムを複数台用意してマルチプロセッサのシステム構成とすれば，故障したシステムを自動的に切り離し，残りのプロセッサで処理を続行していくことができる.

将来

88000にはCMOSでの高速化の計画もあり，マルチプロセッサでの利用とあわせて，超高速性の必要がある応用に利用されていくであろう．また，フォールトトレランスの機能は，今後ますます重要になっていくと思われる．このような方向での利用も広がっていくであろう.



Data Unit中のData Access Pipelineは，2個までのメモリアクセスと1個のアドレス生成を同時に扱え，64bitの浮動小数点数などを1命令で転送する場合にも，命令実行のパイプラインをつまらせることがないようになっている．Instruction Pipelineも2命令を先行して読み込むことができる.

性能	●MC88000，20MHz ●DS8864SP
Integer	14.6
Floating Point	10.8
SPECmark	12.2

モトローラのDS8864SPは，88100を1個，88200を4個搭載している．どちらかというと整数演算が高速である.

3 CISC,RISC CPU基本性能一覧

最後に，今RISCが置かれている状況を考えよう．性能については，グラフを見るまでもなく完全にCISCを圧倒しているといってよい．それでもなおCISCが使われているのは，単にバイナリコンパチビリティの重要さによるものである．今後システムの世代交代が進むに従って，RISCのシェアが伸びていくのは確定的であろう．

キャッシュ技術の発展によって，CPUの性能を制限する要因としてのフォンノイマンボトルネックは，命令フェッチについてはほぼ解消され，データに関しても緩和された．RISCはこの状況を背景に生まれてきたコンピュータで，高速で読み込まれるデータをよどみなく処理する能力が極めて重要である．連続して入ってくる命令やデータを処理するためには，1クロックに1個の命令を実行できなければいけない．このために，必要な機能，不必要な機能を定量的に洗い直して作られたのがRISCである．

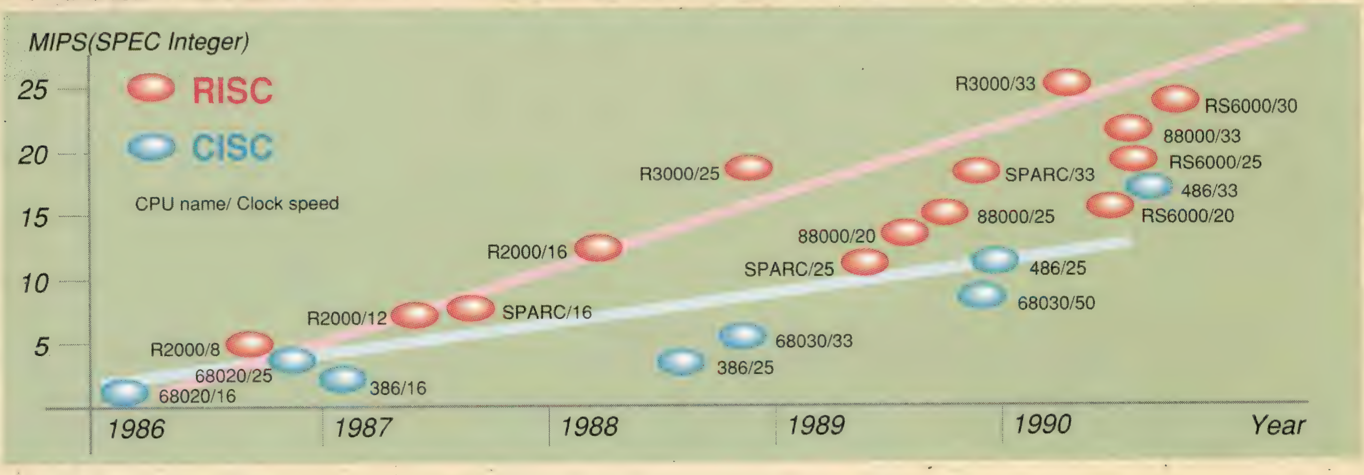
その結果現在のRISCは，下のグラフに見るような総合性能を達成するに至った．しかし，今後コンピュータソフトウェアが発展するのに伴って，コンピュータに要求される能力は，ますます高度なものになっていかざるをえない．ハードウェア技術の発展によってある程度の高速化は達成できようが，CISCからRISCへの発展のようなアーキテクチャ的進歩が今後も必要になるのも間違いあるまい．

ここで発展の方向性として考えられるものの第一に並列化がある．VLIW，スーパースカラーのような粒度の細かい並列化から，密結合マルチプロセッサ，メモリ非共有の粗い粒度の並列化に至るまで，あらゆる手法が考慮されねばならない．CPUの扱かうデータ幅が32bitから64bit，128bitへと拡大されることにより，フォンノイマンボトルネックの解消も重要になる．

このような将来の発展によって，コンピュータはさらに変化していくことになる．すでにCISCを捨て去ったのと同じように，現在のアーキテクチャを否定しなければならない日が必ずくる．

次号では，このようなコンピュータ能力の発展をもたらす，インタラクションの広帯域化について考える．

（編集部）



参考図書

この特集の基点となったのは，やはりこの本である．

■mips RISC Architecture, Gary Kane, Prentice-Hall, 1987

RISCとは何かについて，基本的なアイデアが得られるだろう．アーキテクチャの定量的な評価については，現実のCISC，RISCのほか架空のコンピュータDLXについて評価されているのが，

■Computer Architecture A Quantitative Approach, David A. Patterson and John L. Hennessy, Morgan Kauffmann Publishers, Inc., 1990

洋書しか挙げられないのが残念であるが，このテーマに関してはやむをえまい．最後に，紹介した各プロセッサに関する資料を挙げる．

■SPARC MB86901 Product Description, Fujitsu Microelectronics, Inc., 1988
■IBM RISC System/6000 Technology, Order No. SA23-2619, IBM corp., 1990
■MC88100 RISC Microprocessor User's Manual 2nd edition, Motorola, Prentice-Hall, 1990

宛先

この特集に対するご意見，ご要望，ご指摘などは，こちらの宛先までお寄せください．今後の特集題材の選択についてのご意見もお待ちしております．

〒107-24
東京都港区南青山6-11-1
スリーエフ南青山ビル
株式会社アスキー
スーパーアスキー編集部
特集の係

Windows 3.0の開発環境

Windows 3.0 Software Development Kit

MS-Windows Ver. 3.0（以降Win3）に続き，そのプログラム開発環境であるWindows 3.0 Software Development Kit（SDK）が発表された．Win3の普及が，米国で急速に進んでいるのはご存じのとおりである．プログラマとしては，Win3のプログラム開発環境がどのようなものなのか興味深いところだ．そこで今回のInside Softwareでは，プログラマの視点に立ち，Windowsプログラミングとは何かといった基本的な話題を織り交ぜながら，SDKを解剖してみる．

梅原　系
遠藤　孝信

6000
7500
2000
3000
5000
1000

Windows 3.0SDKの概要

Win3のSDKは，従来のSDKから大幅な改良，機能強化がなされている．ここではまず，SDKの全体像を紹介しよう

パッケージの内容および必要とされるシステム構成

Win3のSDK（以降は単にSDKと略）は，Win3用のプログラム開発で必要になるさまざまなソフトウェアを，パッケージにしたプロダクツである．**表1**に，SDKに含まれるマニュアルや主なツールおよびファイル，価格などをまとめた．表からも分かるとおり，Cコンパイラやアセンブラなどの処理系は含まれていない．これらにはそれぞれ，別売のMS-C，MASMなどを用意する必要がある．また表中に示したように，Win3のSDKに付属するマニュアルと同じものが，Microsoft Press（MSP）から一般の書籍として発売されている．

SDKを利用するために必要なハードウェア／ソフトウェアの一覧を**表2**に示す．開発環境としては，286以上のCPUを搭載したシステムが想定されている．ただし後述するように，386（以上）のCPUに対しては，非常に高度なデバッグ環境が提供される．したがって効率よくプログラム開発を行なうには，386マシンを用意したいところだ．

Win3同様，SDKにもインストールプログラムが用意されている．このインストールプログラムを使えば，SDKをハードディスクにインストールするのは容易である．ただ，残念なことにこのインストールプログラムはキャラクタベースのもので，ウィンドウを使ったWin3のインストールプログラムと比べると，見劣りしてしまう．相手がプログラマとはいえ，Win3のインストールプログラムの完成度が高かっただけに，残念である．

大幅に強化されたデバッグ環境

SDKの目玉となるのは，強力なデバッグ環境である．周知のとおりWindows

は，ノン・プリエンプティブなマルチタスクを実現している．Windowsでは，プロセスごとのアドレス空間が独立していないことやDLLの存在など，デバッグを複雑化する要因が非常に多い．これらの障害を克服するためにも，強力なデバッグ環境が必要とされるのは明らかである．

今回のSDKでは，従来よりも大幅にデバッグ環境が強化されている．その中でも特に注目されるのが，Windows対応のCodeView，CVWだ．MS-DOS（以降DOS）に始まり，OS/2でも提供されているデバッグ環境がWindows用にも提供されたことで，開発環境の共通化が進み，プログラマが新たなツールに戸惑うことなく開発に専念できるようになった．

CVWを含めて，SDKが提供するデバッガはいずれも，デバッグ用画面としてセカンダリモニタ（Secondary Monitor）（もしくはシリアル端末）を使用する．IBM-PCではビデオ回路が本体と独立しているため，１つのシステムで複数のディスプレイボードを使用可能である．これを利用したのがセカンダリモニタだ．セカンダリモニタを使えば，デバッグを行なうプログラムの表示内容を変更することなく，デバッグ作業を進められる．

表３は，SDKに含まれるデバッガの動作環境，デバッガのユーザーインターフェイス，セカンダリモニタ／シリアル端末のサポートなどをまとめたものである．デバッガによって使用するデバイスが異なるため，全デバッガを使用するためには，セカンダリモニタとシリアル端末の両方を用意する必要がある．CVWは，プライマリモニタ（Primary Monitor）にWindows画面，セカンダリモニタにデバッガ画面を表示する．

SYMDEBは，馴染み深いソースレベルデバッガである．CVWのようなテキストウィンドウは表示されないが，基本的なデバッグには十分利用できる．なお，リアルモードに対応しているデバッガはSYMDEBだけである．SYMDEBは，セカンダリモニタ，シリアル端末のどちらでも利用できる．

WDEB386は，386マシン用のデバッガである．386には，“デバッガサポート機能”として，ハードウェア・ブレークポイントを設定する機能がある．

しかしWDEB386では，この機能を利用している気配はない．WDEB386ではソースレベルデバッグは行なえないので，状況に応じてCVWと使い分ける必要があるだろう．

Helpファイル作成ツール

Win3の特徴の１つとして，強力なオンラインヘルプ機能を挙げることができる．Win3の大部分のアプリケーションには，“Help”メニューが用意されている．ユーザーは作業中に分からないことがあれば，このHelpメニューから随時ドキュメントを参照できる．SDKでも，メッセージやAPIなど，Programmer's Referenceがそのままオンラインヘルプになっており，必要な項目を簡単に参照できるようになっている．

Win3のSDKでは，このヘルプファイルを作成するためのツールとして，HC（Help Compiler）が用意されている．HCは，トピックファイル（Tpic File）と呼ばれるRTF（Rich Text Format）*1テキストをWinHelpフォーマットに変換するコンバータである．

このように，ヘルプファイルの作成にはRTF形式のテキストを扱えるエディタが必要になる．しかし，SDKにはRTF対応のエディタは用意されていない．したがって，トピックファイルを作成するには，MS-Wordのようなアプリケーションを別途用意しなければならない．MS-WordにはWin3対応版（WinWord）もあるので，これを利用すればよいだろう．

リソース作成ツール

従来のSDKでは，アイコン，マウスカーソル，ビットマップの作成ツールとして，アイコンエディタ（ICONEDIT.EXE）が提供されていた．これに対しWin3のSDKでは，SDKPaint（SDKPAINT.EXE）というビットマップ編集ツールが提供される．

SDKPaintの編集機能は，Win3本体に付属するPaintBrushのサブセット程度のものだが，異なるディスプレイにおける表示をエミュレートする機能など，PaintBrushにはない機能が多数用意されている．

プログラム最適化ツール

Win3のSDKでは，プログラム最適化ツールとして，Swap TunerとProfilerが新たにサポートされた．

Swap Tunerはセグメント間コールのオーバーヘッド（セグメントの破棄や再ロード）を削減するために，セグメント間コールやセグメントの破棄，ロードなどの記録をとるものである．アプリケーションの実行記録を検討すれば，セグメント間コールの最適化に役立てることができる．

Profilerは，リアルタイムクロックによるインターバル割り込みを利用して，プログラムのどの部分でCPUタイムを消費しているかを計測するものである．CPU時間を消費しているルーチンを集中的にチューンすることで，より効率的にプログラムの高速化を図ることができる．

32bitプログラミングモデル

周知のとおり，286以下のCPUでは，１セグメントのサイズが64Kbytesに制限される．これは，アドレスオフセットのサイズが16bit（$2^{16}=65536$）だという理由による．

しかし386では，アドレスオフセットが32bitなので，理論上CPUの能力としては１セグメントのサイズを2^{32}bytes，4Gbytesの連続したメモリブロックとして扱うことが可能だ．Win3のSDKでは，この32bit専用のAPIを含んだダイナミックリンク・ライブラリ，WINMEM32.DLLが提供される．

この32bitAPIを利用すれば，プログラ

ムは64Kbytesのセグメントに妨げられずに，広大なメモリ空間を使用できる．もちろん32bitAPIを利用できるのは，386／486マシンにおいて，Windowsを386Enhancedモードで実行した場合に限られる．また，WINMEM32.DLLはSDKに付属するもので，ランタイム版のWin3に付属するものではない．したがって，32bitAPIを使ってアプリケーションを作成した場合には，そのアプリケーションとともにWINMEM32.DLLをエンドユーザーに配布する必要がある．

この32bitAPIは，これまで64Kbytesの制限に悩まされてきたプログラマにとっては，願ってもない機能だろう．だが，問題もある．

第1に，CPU自体は32bitであっても，現状のMS-C（Ver.5.1, Ver.6.0）では，32bit長のポインタはサポートされていない．したがって32bitオフセットのメモリをアクセスするには，アセンブラルーチンが必要になる．32bit環境をサポートするCコンパイラの出現が待たれるところだ．

第2の問題点は，Windowsカーネルが16bitオフセットを前提に設計されている，ということだ．したがって，APIなどを使用する場合（WINMEM32.DLLを含む），16bitに変換するインターフェイスを用意しなければならない．

第3の問題は，すでに述べたように，32bitAPIは386Enhancedモードでしか利用できないことだ．このためReal, Standardモードでも動作可能なアプリケーションを作成するには，起動時にWindowsの動作モードを調べ*2，モードに応じて実行するコードを変える必要がある．

このようにWin3の32bitプログラミングモデルは，まだまだ不完全なものであり，利用に当たっては十分な配慮が必要だ．OS/2 Ver.2.0では，より完全なサポートが行なわれる予定である．しかし，そのOS/2よりも先にWin3が32bit環境をサポートしたことは，重大な意義があると思われる．第3の問題点として，32bitAPIが386Enhancedモードでしか利用できないことを挙げたが，これは逆に，386でしか動作しないアプリケーションがリリースされる要因ともなる．今後の市場の動向に注目したい．

カラーパレットのサポート

現存するパーソナルコンピュータの多くは，ビデオ表示回路の一部として，カラーパレットレジスタを備えている．ところが従来のWindowsでは，カラーパレットの存在を無視していたため，表現力が大きく制約されていた．

Win3では，カラーパレットをシステムが管理することで，より多彩な表示を可能にしている．カラーパレットを使用しても，同時に表示できる色数は増えないが，任意の中間色が利用できることで，表現力ははるかに高まる．

カラーパレットをWindowsが管理するようになったことで，アプリケーションのデバイス独立を実現すると同時に，表示デバイスの性能を引き出すことが可能になった．Windowsではグラフィックを扱うアプリケーションが多いだけに，この機能も存分に利用されるだろう．

（梅原　系）

＊1　Microsoftが提唱した，テキストファイルフォーマット．Microsoftが発表しているワードプロセッサでは，RTFフォーマットのファイルを出力するようになっている．

＊2　このためのAPIとして，GetWinFlagsがある．

表1　SDKの構成

製品名	Microsoft Windows Software Development Kit for Version 3.0	
発売元	Microsoft Corporation（米）	
価格	500ドル（アップデートの場合は150ドル）	
付属マニュアル	Guide to Programming　Programmer's Reference　Programming Tools （以上3点はMicrosoft Pressより書籍として別売されている） Installation and Update Guide　IBM CUA Style Guide	
主なツール	デバッガ	CodeView for Windows（CVW） Symbolic Debugger（SYMDEB）　386 Debugger（WDEB386）
	デバッグ補助ツール	Spy　Heap Walker　Shaker
	リソースエディタ	SDKPaint　Dialog Editor　Font Editor
	リソースコンパイラ	Resource Compiler(RC)
	Helpファイル作成ツール	Help Compiler(HC)
	最適化ツール	Swap Tuner　Profiler
主なファイル	ドキュメント オンライン・ドキュメント インクルードファイル ライブラリ デバッグ用カーネル サンプルソース	WinHelp用　Adviser用

表2　必要とされるシステムの構成

ハードウェア	
CPU	286／386（SX）／486
メモリ	コンベンショナルメモリ　640Kbytes 拡張メモリ　384Kbytes以上
FDD	1基
HDD	最低4.2Mbytes（SDKのみ） セカンダリモニタ（デバッガ使用時） シリアル端末（デバッガ使用時）
ソフトウェア	
DOS	Ver.3.1以上
Windows	Ver.3.0以上
Cコンパイラ	MS-C Ver.5.1以上またはQuick-C Ver.2.0以上

表3　デバッガの動作環境その他一覧

	CVW	SYMDEB	WDEB386
動作環境*	S／E	R	S／E**
シンボリックデバッグ	Y	Y	Y
ソースレベルデバッグ	Y	Y	N
テキストモードウィンドウ	Y	N	N
セカンダリモニタ	Y	Y	N
シリアル端末	N	Y***	Y

［注］
*　S…Standardモード／E…386Enhancedモード／R…Realモード
**　386マシンのみ
***　出力デバイスとしてのみ使用（キーボードは無効になる）

Windowsプログラミングとは何か

Windowsのプログラミングとはどのようなものなのか．ここでその実際を探ってみよう

マルチウィンドウや，一貫したGUIのための API (Application Programming Interface)など，Windowsはこれまでにはなかったさまざまなサービスをプログラムに提供する．しかしその一方では，プログラミングスタイルが従来と大きく異なるため，Windowsプログラミングが手の届きにくいものになっているのも事実だ．Windowsプログラミングでは，DOSプログラミングにはなかったさまざまなモジュールファイルを作成する必要がある．ここでは，Windowsプログラムを構成するモジュールファイルを個々に取り上げ，その役割を述べる．続いて，これらのファイルが，どのような過程を経てWindowsプログラムになるのかを追ってみる．

プログラムソースファイル(.Cファイル)

Windowsプログラムの作成には，C言語，アセンブラを利用する．独自にインターフェイスを作成すれば，たとえばBASICコンパイラなどを利用することも可能だが，現状ではあまり一般的ではない．実際，SDKに付属するインクルードファイルやライブラリなどは，すべてCコンパイラ（もしくはアセンブラ）を前提としている．

良質なコンパイラが利用できるようになったことなどから，現在ではパーソナルコンピュータにおいても，ハードウェアドライバなどのシステムプログラムからアプリケーションに至るまで，C言語を使ってプログラミングすることが一般化している．

こうした背景を考えれば，Windowsプログラムの開発にC言語を使うこと自体に違和感はないだろう．しかしWindowsのプログラムコードは，通常のCコーディングとは若干趣が異なっている．これには次のような理由がある．

■プログラムコードの多くが，WindowsのAPIコールで構成される*3．
■Windowsは，Cプリプロセッサの#typedefを使用して，Windows独自の型を再定義している．
■Windowsプログラムでは，変数の命名法としてハンガリー記法（Hungarian notation）*4を推奨している．

Windowsプログラミングを始めたばかりのときには，この“Windows独自の”コーディング方式に戸惑うかもしれない．しかしこれらはどれも，プログラムコードを読みやすく，論理的な組み立てを分かりやすくするためのものである．いくつかプログラムを作っているうちに，自然と身に付くことだろう．

モジュール定義ファイル(.DEFファイル)

Windowsでは，プログラムを構成するコードセグメントおよびデータセグメントのそれぞれに，移動可能，破棄可能といった属性を付けることができる．ここでは詳細は述べないが，Windowsは，この情報をもとにセグメントを移動したり破棄したりしながら，物理メモリをダイナミックに管理している．

このセグメントの属性をリンカに通知するのが，モジュール定義ファイル（拡張子.DEF)である．このほかにもモジュール定義ファイルでは，ローカルヒープのサイズやスタックのサイズなどを指定する．リンカは，このモジュール定義ファイルで指定された情報をもとに，実行モジュールを生成する．

なおVer.2.Xでは，LINK4.EXEというWindows専用のリンカがSDKで提供された．これに対しWin3では，MS-Cに付属するLINK.EXEをそのまま利用できる．

リソース記述ファイル(.RCファイル)

通常のCプログラムにおいて，何らかのテキストを画面に表示するには，たとえば次のようにするだろう．

```
printf(“Hello World¥n”);
```

そしてこの“Hello World”という文字列は，プログラムのデータセグメントに格納される．

これに対しWindowsでは，こうしたテキストなど，プログラムが使用するデータを，リソース（Resource：資源）として通常のデータセグメントとは区別して実行モジュール中に格納しておくことができる*5．

Windowsが扱えるリソースの一覧を**表4**に示す．これらのリソースは，データセグメントではなく，リソースセグメントと呼ばれるセグメントに格納される．プログラムの作成時に，こうしたデータをリソースとして定義しておけば，プログラムを再コンパイルすることなく，データだけを修正することが可能となる．たとえば，プログラムが使用するテキストがすべてリソースになっているなら，外国語に対応させる場合でも，リソースデータを修正するだけでよい．

リソースを定義するには，プログラムソース同様，エディタでリソース記述ファイルというASCIIファイルを作成する．このとき，拡張子は.RCにしておく．.RCファイルをSDKに付属するリソースコンパイラ（RC.EXE）でコンパイルすると，バイナリ形式の.RESファイルが生成される．リソースを.EXEに組み込むには，この.RESファイルを指定して，再度RCを起動すればよい．

なお，.RCファイルはASCII形式だが，ビットマップなどをバイナリ形式のファイルとして作成しておき，.RCファイルで取り込むことも可能である．

インポートライブラリ

これまでに述べたプログラムソース(.C)，モジュール定義ファイル(.DEF)，リソース記述ファイル(.RC)は，いずれもプログラマが作成しなければならないものである．しかしこれ以外にも，Windowsプログラムには欠かせないファイルがあるのでここで紹介しておこう．それは，インポートライブラリ(Import Library)である．

ご存じのようにWindowsでは，ダイナミックリンクと呼ばれる機構を使って，プログラムとWindowsのAPI関数を実行時に動的にリンクしている．実行時にリンクが完了するということは，.EXEファイルを生成した時点では，解決されない外部参照が残っていることになる．これは，通常のDOSプログラムのようなスタティックリンク*6では考えられなかったことだ．

インポートライブラリには，WindowsのAPI名，そのAPIの実体が存在するWindowsのモジュール名*7，そのAPIの関数番号が一組となり，これがAPIの数だけ格納されている．リンカは，プログラム中のAPIコールに対して，そのAPIが含まれるモジュール名，そのAPIの関数番号をインポートライブラリから探し出し，NewEXEヘッダにその情報を書き込む．そしてプログラムが実行されると，WindowsはNewEXEヘッダの情報をもとに，APIの実体とプログラム中のAPIコールをリンクするのである．

＊3　たとえば，メッセージを取得するAPIはGetMessageという名称である．このようにWindowsのAPIは，大文字と小文字によって構成され，Cの標準ライブラリに比べると，一般にその名称も長い（IsClipboardFormatAvailableというAPIもある)．APIの名称からその機能を連想しやすいのはよいが，必然的にコーディング時のタイプ量は多くなる．

＊4　変数名の直前に変数の型(charなら'c'，intなら'i'など）を付け，変数名からその型が分かるようにする命名法．Microsoftのプログラマ，チャールズ・シモニー(Charles Simonyi）が考案した．

＊5　もちろん，リソースを使用しないで，従来どおりにデータセグメントに格納することもできる．

＊6　プログラムが使用するライブラリ関数のコードを，すべて.EXEファイル中に抱えるリンク方式．

＊7　KERNEL，USER，GDIなど．

実行モジュールの生成

さて，Windowsプログラムを構成するファイルの種類，その働きが分かったので，次は実際のプログラム開発がどのように進むのかを述べよう．

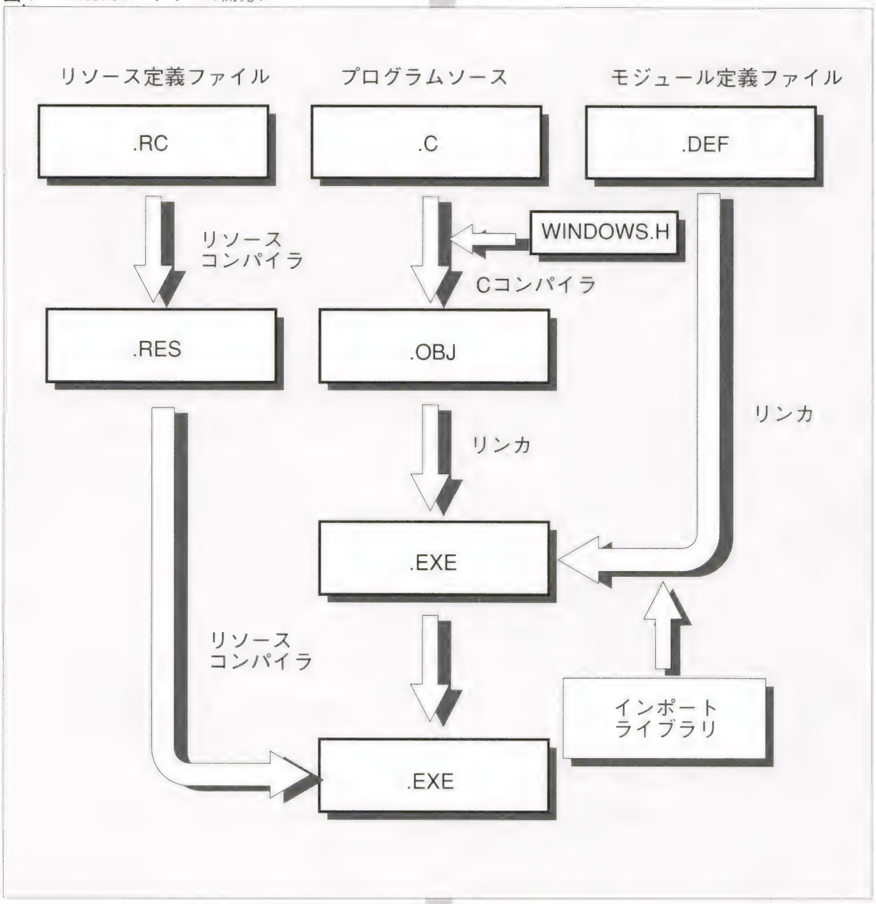
図1に，Windowsプログラムの開発フローを示す．図中には示されていないが，通常はMAKEユーティリティを使って，これら一連の処理を行なうことになるだろう．

DOSプログラムよりも多くのファイルを作成しなければならないので，面倒だと思った読者がいるかもしれない．しかしリソース定義ファイル(.RC)やモジュール定義ファイル(.DEF)は，一度作成したら変更することはさほどない．したがってWindowsプログラムのデバッグは，.Cプログラムの変更，コンパイル(.OBJの生成)，リンク(.EXEの生成)，.EXEへのリソース（.RESファイル）の組み込み，という工程を繰り返すことになる．.RCファイルが変更されていなければ，リソースコンパイラを起動して，.RESを再生成する必要はない．(編集部)

表4　Windowsで利用可能なリソース

アイコン	ダイアログボックスの内部などに表示されるビットマップ
カーソル	マウスカーソルの形状
ビットマップ	任意のサイズのビットマップ
文字列	表示に使用する文字列
メニュー	メニュー項目の定義
キーボードアクセラレータ	ショートカットキーの定義など
ダイアログボックス	ボタンなどのコントロールを含むウィンドウ
フォント	文字フォント
ユーザー定義リソース	プログラマが独自に定義できるリソース

図1　Windowsプログラムの開発フロー



Windowsのメモリ管理

Windowsプログラムでは，効率よくメモリを扱うことが重要だ．ここでは，Windows一般のメモリ管理について述べる

Windowsでは，複数のプログラムが同時に実行されるため，あるプログラムがメモリをどのように使用するかは，そのプログラムだけでなく，システム全体のパフォーマンスに大きな影響を及ぼす．Windowsプログラマは，Windowsが内部で行なっているメモリ管理を理解し，適切な方法でメモリを扱う必要がある．

ここではまず，従来（Ver.2.X）のWindowsのメモリ管理方法を述べ，続いてWin3のStandard，386Enhancedモードのメモリ管理について述べることにする．今さら従来の方法を知る必要はないと思うむきがあるかもしれない．しかしWin3のRealモードでは，Ver.2.Xと同様の方式でメモリ管理を行なっている．さらにStandard，386Enhancedモードにおいても，互換性の問題など，従来の方式を理解することなしには語れない部分も多い．

Windowsに必要とされるメモリ管理機能

Windowsは，DOS環境で動作するオペレーティング環境である．しかしDOSは，ご存じのとおり，ファイルI／Oやファイルローダの機能を備えているにすぎない．したがってWindowsは，マルチウィンドウ，マルチタスクを実現するために，本来オペレーティングシステムが用意すべきメモリ管理などの機能を自分自身で実現しなければならない．

Windows Ver.2.Xは，8086CPUが管理できるメモリ（通常は640Kbytes）の中で基本的な動作を行なう必要があった．実際には，この640Kbytesにデバイスドライバや他の常駐モジュールが組み込まれ，さらにWindowsのシステムが常駐するので，アプリケーションが使用できるメモリ領域はわずかしか残らない．ある意味でWindows開発の歴史は，“狭いメモリ領域”との闘いだったといえるかもしれない．Windowsは，この“狭いメモリ領域”を最大限に有効利用するため，次のような機構を備えている．

セグメントの共有

初めに述べたとおり，Windowsプログラムは，基本的にC言語を使って作成することになる．ご存じのようにC言語では，プログラムのコード部とデータ部が明確に分離される．そしてコード部にはプログラムコードが格納され，データ部にはプログラムのデータが格納される．ここで重要なのは“コード部に格納された内容は，プログラムの実行時も変更されない”ということだ．DOSでは，このコード部とデータ部を分離するために，80X86のセグメントを使用している．

複数のプログラムが同時に実行されるWindowsでは，このように常に内容が変更されないセグメント（コードセグメント）をそれらのプログラムで共有できれば，非常に効率がよい．特にWindowsアプリケーションは，その処理の多くをWindowsから提供されるDLL呼び出しに当てている．セグメントの共有ができれば，プログラムが多数実行された場合でも，DLLは常に1組だけメモリに存在すればよいことになる．ここでは，説明を簡単にするためにコードセグメントを例にとったが，内容を変更されないことが保証されたデータセグメントを，複数のプログラムで共有することも可能である．

セグメントの破棄と再ロード

WindowsのAPIは，500種類以上にも及ぶ．セグメントを共有できるのはよいが，大規模なアプリケーションを実行した場合，1組のDLLをメモリにロードできるという保証すらない．

ここで注目すべきは，プログラムの局所性である．ご存じのように，ある一瞬に着目すれば，プログラムが必要とするメモリはごく限られたものでしかない．これは，たとえそれがマルチタスク環境であっても変わらない．ある瞬間に実行されるのは，あるプログラムの，ある一部のコードである．すべてのプログラムコードが，常にメモリに存在する必要は必ずしもない．

これを具体的にいえば，プログラムを起動する場合でも，そのすべてのコードセグメントをメモリにロードする必要はないということだ．また，コードセグメントの内容は変更されないのだから，1度メモリにロードされたコードセグメントでも，当面必要でなければメモリから削除（破棄）してしまっても，再度アクセス要求が起こった時点で，再びディスク内の実行モジュールからロードし直すような機構があればよいことになる．

これを利用したのが，Windowsにおけるメモリの破棄と再ロードである．Windowsは，物理メモリが足りなくなると，LRU（Least Recently Used）アルゴリズム[*8]によって，セグメントを破棄する．その後，破棄されたセグメントが必要になったら，それをディスクから再度ロードし直すのである．なお，ここでもコードセグメントを例にとって説明したが，データセグメントを破棄することも可能である．

ダイナミックなセグメントの移動

個々のセグメントサイズはまちまちなので，セグメントの破棄と再ロードを繰り返すと，物理メモリに多数の未使用領域のフラグメント[*9]が発生してしまう．そのため，未使用のメモリ空間全体の大きさが十分でも，プログラムをロードできるだけの連続した空きメモリを確保できない場合がある．

すでにメモリの破棄については述べた．

だが実際には，Windowsはメモリを破棄する以前に，セグメントを移動して，フラグメントを連続したメモリブロックにする．メモリの破棄が実際に起こるのは，セグメントを移動してフラグメントを集めても，なおメモリが不足する場合である．このようにWindowsは，プログラムの実行時に，セグメントをダイナミックに移動している．

BURGERMASTER

これまでに何らかのDOSプログラムを作った経験がある読者は，これまでの説明に驚かれたかもしれない．Windowsでは，1度参照したメモリブロックを再度アクセスする場合でも，それが以前と同じアドレスにあるとは限らない．いや，メモリにない場合すらあるのだ．

こうしたメモリ管理機構を構築するのは容易ではない．実際には，メモリの移動や破棄によって起こる不都合もたくさんある．そこで次は，そうしたメモリ管理手法をとろうとした場合に起こる不都合とは何なのか，Windowsがどのようにしてその問題を回避しているのかについて述べよう．

いまプログラムで，あるメモリブロックをシステムから取得し，使用する場合を考えよう．普通に考えれば，メモリブロックのアドレスを取得し，そのアドレスに対してデータの読み書きを行なうことになる．しかしWindowsは，セグメントをダイナミックに移動するので，メモリブロックが存在するアドレスは固定ではない．

このためWindowsプログラムは，メモリブロックを取得する場合でも，アドレスではなく，ハンドルという値をWindowsから取得する．その後プログラムでは，このハンドル値を使ってメモリブロックにアクセスする．

当然のことながら，ハンドル値を使ったメモリ参照は，最終的にはセグメントアドレスに置き換えられなければならない．このためWindowsは，ハンドル値と実際のセグメントアドレスを対応させた一覧表を管理している．この一覧表が存在するセグメントは，BURGERMASTERと呼ばれる．

Windowsは，セグメントを移動すると同時に，BURGERMASTERに記述されたセグメントアドレス値を更新する．そしてプログラムでは，メモリブロックを参照する直前にハンドル値をWindowsに与えて，実際のセグメントアドレスを取得するのである(具体的には，Global LockというAPIを呼び出す)．

再ロードサンク

Windowsがコードセグメントを破棄したり，再ロードしたりしていることはすでに述べた．このセグメントの破棄と再ロードを行なうために，Windowsは再ロードサンク(Reload Thunk)という小さなメモリ領域を管理している．

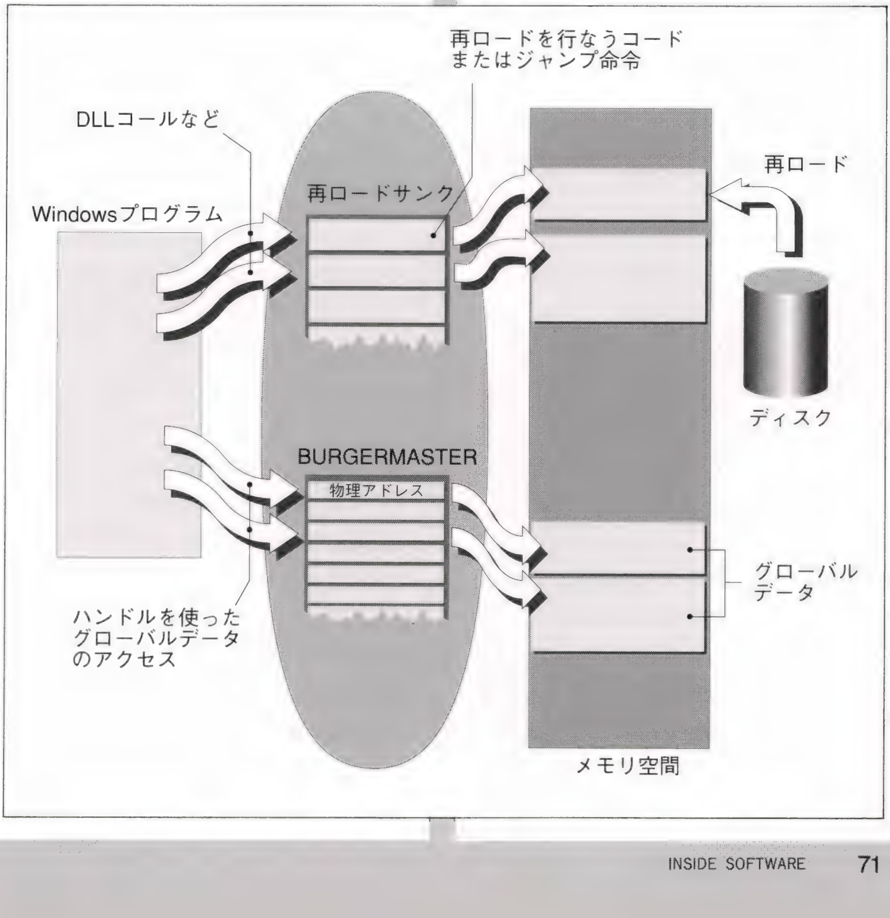
再ロードサンク内部には，目的のコードセグメントが現在メモリにあるかどうかにより，次の値のどちらかが格納されている．

1．セグメントがメモリ中にある場合は，そのコードへの単純なジャンプ命令
2．セグメントがない場合は，そのコードをディスク中のファイルからロードするためのコード．

DLLを呼び出す場合など，Windowsプログラムからのfarコールは，実際にはこの再ロードサンクを呼び出すようになっている．Windowsは，セグメントを破棄したら，再ロードサンクに格納されているジャンプ命令を2.の内容に変更する．

BURGERMASTERと再ロードサンク，Windowsプログラムの関係を**図2**に示した．このようにWindowsは，本来OSが担うべき役割を，自分自身で行なっているのである．

図2　BURGERMASTERと再ロードサンク，Windowsプログラムの関係



＊8　その時点で，最も長い間，アクセスされていないセグメントを特定するためのアルゴリズム．
＊9　不連続に分断されたメモリの断片．

Win3におけるメモリ管理の実際

Win3は，286，386のプロテクトモードをどのように利用しているのか．その実際を探ってみる

Win3は，286，386のプロテクトモードに対応することにより，640Kbytesの壁を破ることに成功している．しかし，単に640Kbytes以上のメモリを扱えるという意味では，従来でもEMSのサポートがあった．それでは，プログラマにとって，Win3がプロテクトモードに対応したというのはどういうことなのか．

Win3の発表後，米国ではOS/2の今後を危ぶむ声も多い．これからのオペレーティング環境を占ううえでも，Win3の大きな特徴の1つであるメモリ管理能力を理解しておくのは重要なことだろう．

リアルモードとプロテクトモード

リアルモード*10では，アドレス指定にセグメントアドレスとオフセットアドレスを使用する．これらはそれぞれ16bit幅で，セグメントアドレスを左に4bitシフト(16倍)し，それにオフセットアドレスを加えることで，1Mbytes（2^{20}＝1048576 bytes）の物理アドレス空間を利用できる．

これに対しプロテクトモードでは，セグメントアドレスの代わりにセグメントセレクタを指定する．セグメントセレクタは，セグメントアドレスが記述されたテーブルへのインデックスになっている．このテーブルは，ディスクリプタテーブルと呼ばれる．セグメントディスクリプタには，セグメントアドレス以外にも，セグメントの属性（実行可能，書き込み可能など）やセグメントの特権レベルなどが格納されている（**図3**）．

実際プロテクトモードには，マルチタスクOSを意識した機構が数多く導入されている．ここでは誌面の都合で詳細は述べないが，ポイントは次のような点だろう．

■アドレスの上位16bit（リアルモードでセグメントアドレスに当たるもの）がセグメントディスクリプタを指すインデックスになっている．

■セグメントディスクリプタには，セグメントアドレス以外にも，セグメントの属性その他を格納する領域がある．

さて，この点を踏まえて，まずはWin3のStandardモードを眺めてみよう．

Win3のStandardモード

Standardモードにおける最大のメリットは，1Mbytesを超えるメモリ空間を主記憶としてアドレッシングできることである．従来もEMSを使って1Mbytes超のメモリをアクセスすることは可能だったが，これはあくまでも主記憶ではない．プログラムからこのメモリをアクセスするには，複雑なEMSの手続きを踏まなければならなかった．

Standardモードでは，EMSなどに頼ることなく，Windowsが提供するAPIを使って，すべての物理メモリをアクセスすることが可能である．ただし誤解してはならないのは，セグメントオフセットのサイズは従来と同じ16bitだということだ．1Mbytes超のメモリを扱えることは事実だが，あくまでも1セグメントのサイズは64Kbytesである．64Kbytes以上のデータを扱うためには，MS-Cのヒュージポインタを使用したり，プログラムをヒュージモデルでコンパイルしたりしなければならない．リアルモードでは，プログラムで“次の”セグメントアドレスを算出し，セグメント値を直接指定して，64Kbytes以上のメモリを参照することも可能だった．しかしプロテクトモードでは，セレクタの値をプログラムで算出して，64Kbytes以上の物理メモリを参照するようなことはできない．これを行なうと，GPフォルト*11が発生するか，最悪の場合は予想外のセグメントの内容を破壊してしまうこともある．

プログラムを作成するうえでは直接関係ないことだが，ここでStandardモードで物理メモリがどのように利用されているのかを簡単に述べよう．Standardモードでは，通常，物理メモリを次の3つの領域に分けて管理している．

■コンベンショナルメモリ(Conventional Memory)
1Mbytes以下，つまりリアルモードで参照可能なメモリ領域．この領域には，TSRプログラム，デバイスドライバ，MS-DOSシステムのほか，リアルモードから参照できる必要のあるデータ（PSPブロックやRS-232C用のデータキューなど）が格納される．

■拡張メモリ（Extend Memory）
1Mbytes超の主記憶領域．Win3は，XMSドライバ（HIMEM.SYS）を通じてこの拡張メモリ領域を獲得する．ただし，その後のメモリアクセスはXMSドライバを経由せずに，直接行なっている．

■HMA（High Memory Area）
286，386のA20アドレスラインをアクティブにすることによってアクセス可能な1Mbytes超のメモリ領域（65520bytes）．Win3は，その起動時に可能であれば，HMAをWin3が管理する領域として認識する．Win3の起動以前に，何らかのプログラムがHMAを利用している場合は，Win3の管理領域にはならない．HMAも，XMSドライバ経由で制御される．

さらばBURGERMASTER

Windows Ver.2.Xでは，プログラムからのメモリ要求に対してハンドル値が返され，ハンドルはBURGERMASTERというテーブルを参照すると述べた．そしてWindowsがセグメントを移動したり，破棄したりするときには，BURGERMAS

TERのエントリを変更する．この機構により，プログラムではハンドル値を使って正しいセグメントを参照することが可能となる．

セグメントセレクタとセグメントディスクリプタを使ったプロテクトモードのメモリ管理手法を見ると，BURGERMASTERが果たしていた役割は，このプロテクトモードの機構をシミュレートすることだったと分かる．つまり，プロテクトモードにおけるセグメントセレクタ値が，ハンドル値に当たるわけだ．

したがってStandard，386Enhancedモードでは，もうBURGERMASTERは必要ない．BURGERMASTERの役割は，CPUがハードウェアで処理してくれる．実際，Win3でプログラムがメモリを取得すると，セレクタ値が返される．

Win3の386Enhancedモード

Standardモードのメモリ機構に，386のページング機構，仮想86モードの機能を追加したのが386Enhancedモードのメモリ管理方式である．

■386Enhancedモードでは，拡張メモリ領域を広大な１つの仮想空間として扱う．Standardモードのように，HMAとそれ以外の領域を区別しない．
■仮想空間の大きさは，物理メモリのサイズのみによっては決定されない．386Enhancedモードでは，ページング機構によってディスクを第2のメモリとみなすことができる．

メモリの移動や破棄といった機構は386Enhancedモードにもあるが，実際の動作は他のモードとはかなり異なっている．Real，Standardモードでは，物理メモリが不足した場合，セグメント単位にメモリの移動や破棄が行なわれる．これに対し386Enhancedモードでは，物理メモリが不足すると，メモリの移動や破棄を行なうのではなく，386のページング機構によって4Kbytes単位に物理メモリの内容をディスクにスワップアウトする．仮想空間がなくなるまで，Win3はメモリの破棄を行なわない．

こうした処理は，プログラムからはまったく透過に行なわれる．プログラムがディスクにスワップアウトされたメモリブロックにアクセスすると，386がページフォールト（Page Fault）と呼ばれる例外(割り込み)を発生する．するとWindowsは，そのページをディスクから読み出し，物理メモリにマッピングする．

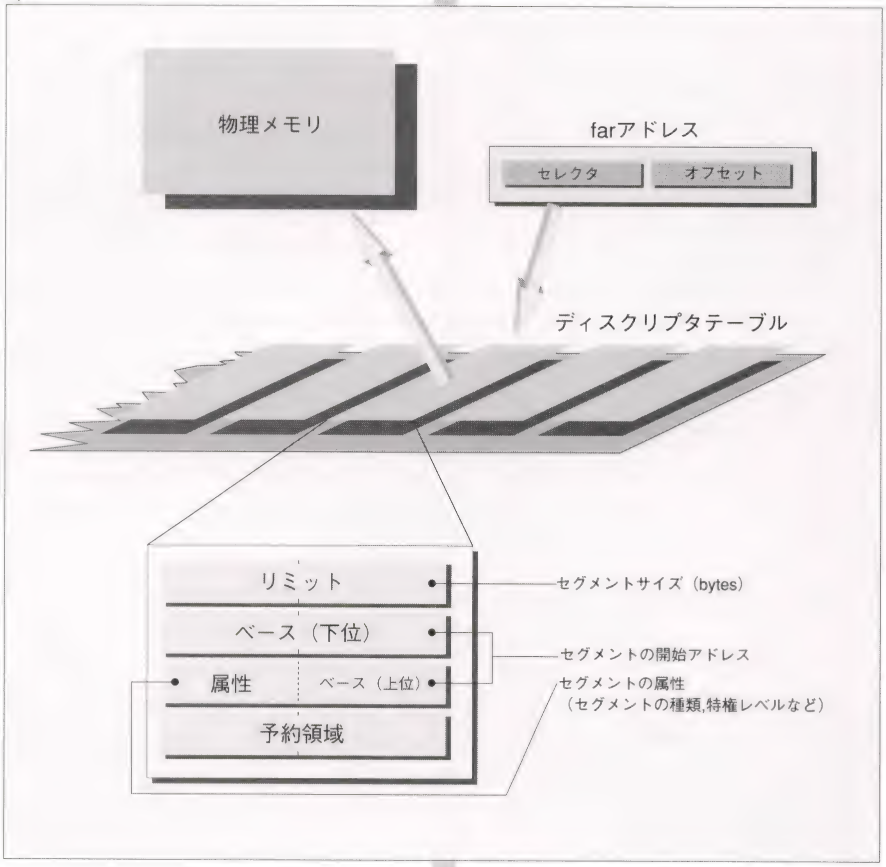
386Enhancedモードでも，１セグメントのサイズが64Kbytesであることは変わらない．初めに述べたとおり，SDKに付属するWINMEM32.DLLを使えばこの問題を回避できる．しかしその場合には，アセンブラルーチンを独自に用意しなければならない．現状では，64Kbytesを超えるメモリを扱うのは容易ではない．

パーマネント・スワップファイル

386Enhancedモードでは，スワップアウトされたデータがスワップファイルとしてディスクに保存される．このスワップファイルのサイズは，物理メモリとディスク中の空き領域の大きさによって決定される．

しかしSwapFileツール（SWAPFILE.EXE）を使えば，パーマネント・スワップファイルと呼ばれる固定サイズのスワップファイルを作成することが可能だ．SwapFileでは，ディスクで連続している最大の空き領域をパーマネント・スワップファイルとすることができる．パーマネント・スワップファイルを作成すれば，スワップ動作時もヘッドのシークなどを最小限に抑えられるので，高速な処理が期待できる．

図３　セグメントセレクタとセグメントディスクリプタ



＊10　ここでは，80X86CPUの動作モードを指す．Win3の動作モードは，“Realモード”と表記している．

＊11　General Protection Fault．保護規則違反エラー．286以上のCPUでは，セグメントの不正なアクセスなどが行なわれると，GPフォルトが発生する．

デバイスに依存しないカラー環境

Win3では，デバイス非依存の思想がさらに推し進められている．ここでは，Win3のカラー環境を探ってみよう

DIB (Device Independent Bitmap)

Windowsには，GDI (Graphics Device Interface) というグラフィックス・プログラムインターフェイスが備えられている．このGDIでは，表示デバイス(ディスプレイなど) に依存せずにグラフィックス出力が行なえるように，さまざまな機能が提供されている．このようにデバイス非依存を強く意識して設計されたWindowsだが，Ver.2.Xでは，この点で重大な問題が１つあった．

Windowsでは，ビットマップを独立したファイル(.BMP)にしておき，それをリソースとして参照できることはすでに述べた．問題とは，Ver.2.Xでは，このビットマップのデータが，特定のデバイスに依存してしまうということである．したがって，ある表示デバイス用に作成されたビットマップは，VRAMの構成がそれと異なるデバイスでは，正しく表示されない．Ver.2.Xに付属するPaintでモノクロビットマップしか作成できないのは，こうした理由によるものだと思われる．

Win3では，この問題を回避するため，従来とはまったく異なったビットマップフォーマットを採用している．これがDIBである．これと同時に，Win3に付属するPaintBrushでは，カラービットマップを作成できるようになった．

Ver.2.Xのビットマップフォーマット

まずは，従来のビットマップフォーマットを見てみよう．**表５**にVer.2.Xにおけるビットマップファイルのフォーマットを示した．表中には，BITMAP構造体*[12]の対応するフィールド名も同時に示してある．このうち分かりにくいのは，bmPlanesとbmBitsPixelだが，これらはVRAMの物理的な構成を定義する領域である*[13]．表から分かるとおり，Ver.2.Xのビットマップフォーマットでは，ビットマップのサイズ(bmWidth, bmHeight, bmWidthBytes)，ディスプレイデバイスの構成(bmPlanes, bmBitsPixel)が記述されているにすぎない．これでは，色に関する具体的な情報は定義されない（たとえば，ディスプレイのプレーン数がbmPlanesから分かったとしても，それぞれのプレーンがどの色を表わすのかは分からない)．したがって構成の異なるVRAM (たとえば，プレーンのRGB順序が異なるVRAM) では，ビットマップは正しく表示されないことになってしまう．

DIBフォーマット

これに対し，Win3のDIBフォーマットは**表６**のようになっている．項目を参照しやすいように，**表５**同様に構造体のフィールド名を列挙した．ただし実際には，これらは４種類の構造体によって定義されている．その詳細については，SDKに付属するWINDOWS.Hヘッダファイルを参照していただきたい．

一見しただけでも，ビットマップフォーマットの構造が大きく変化していることが分かる．それでは，その詳細を追ってみよう．

DIBフォーマットは，全体として大きく３つの領域に分かれている．第1の領域はbfTypeからbfOffBitsまで，第2の領域はbiSizeからbiClrImportantまで，そして第3の領域はrgbBlueからrgbReservedまでである．

第１の領域は，ビットマップファイル全体に対する情報になっている．ここでは，ビットマップファイルのサイズ(bfSize)，ビットマップ本体へのオフセット (bfOffBits) が定義される．

第２の領域は，ビットマップのサイズや，ビットマップデータの構成を定義する．このうち重要なのは，biPlanesとbiBitCountの意味である．すでに述べたとおりVer.2.Xでは，biPlanesに相当するフィールド (bmPlanes) で物理ディスプレイのプレーン数を定義していた．しかしWin3のDIBでは，この値が常に1に固定されている．そしてDIBでは，biBitCountが１，４，８，24のいずれかの値をとる．この値によりビットマップが表現できる色数は，２(2^1)，16(2^4)，256(2^8)，16777216(2^{24}) 色となる．これらはそれぞれ，モノクロビットマップ，16色ビットマップ，256色ビットマップ，24bitビットマップと呼ばれる．

第３の領域は，カラーテーブルである．テーブルの１エントリは４bytesで，そのうちの１byte (rgbReserved) は予約領域になっており，現時点では使用されない．

DIBフォーマットが従来のフォーマットと根本的に異なるのは，ビットマップイメージ（ビットマップ本体）が物理的なVRAMを意識したピクセルイメージではなく，カラーテーブル (第3の領域) のインデックスになっているということだ．biBitCountの値が１，４，８の場合，それぞれ２，16，256エントリのカラーテーブルが作成される．ただしbiBitCountが24の場合は例外で，このときにはカラーテーブルは作成されずに，ビットイメージが直接RGB値 (カラーテーブルにおけるrgbBlue, rgbGreen, rgbRedに相当する値) を持つようになる．

DIBでは，ビットマップを圧縮してファイルに保存することもできる．データが圧縮されているかどうかは，biCompressionの値を調べればよい．biCompressionは，RI_RGB, RI_RLE8, RI_RLE4のいずれかの値をとる(それぞれ，WINDOWS.Hで０，１，２と定義されている)．この値がRI_RGBなら，ビットマップは圧縮されていない．一方，RI_RLE8, RI_RLE4な

ら，ビットマップはRun-Length Encording (RLE)[*14]アルゴリズムによって，それぞれ8bitあるいは4bit単位の圧縮が施されていることを示す．圧縮後のイメージのサイズは，biSizeImageに格納されている．

biYPelsPerMeterおよびbiXPelsPerMeterには，それぞれデバイスが1メートル当たりに表示できるy方向，x方向のピクセル数が格納されている．この値を利用すれば，ビットマップを正確に表示することが可能となる．

パレットマネージャ

当然のことながら，出力デバイスが同時に表示できる色数には制限がある．たとえば，前述の24bitビットマップでは，理論的には16777216 (2^{24}) 色を1つのビットマップで使用できることになるが，実際にはこれを同時発色できるディスプレイは一般的ではないだろう（ちなみに，VGA，EGAでは16色）．それどころか，たとえ16色ビットマップであっても，複数のプログラムが同時に異なるビットマップを表示しようとした場合，それぞれを本来の色で表示できない場合すらある．

このようにWindowsでは，出力デバイスが表示できる色と，個々のプログラムが要求する色をうまく調停する機構が必要となる．このためにWin3で新たにサポートされたのが，パレットマネージャである．パレットマネージャの働きによって，個々のプログラムは独自のカラーセット（カラーパレット）を持てるようになった．プログラムは，ハードウェアの発色能力を意識することなく，自分のカラーパレットに定義した色を使って描画を行なえばよい．この描画要求とハードウェア能力とのギャップは，パレットマネージャが埋めてくれる．

論理パレットとパレットマネージャ

Win3では，個々のプログラムが論理パレットと呼ばれる独自のカラーパレットを持つことができる．一方，この論理パレットに対して，Windowsが管理するシステムの物理的なカラーパレットは，システムパレットと呼ばれる．

論理パレット内のエントリ数は，プログラムで任意に設定できる．そして，それぞれのエントリには，R (Red)，G (Green)，B (Blue) の色の強度を0から255までの整数値で指定する．実際のプログラムでは，まず論理パレットを作成し，カラーエントリを設定してから，出力デバイスを管理するデバイスコンテキスト[*15]でその論理パレットを選択すればよい．するとそれ以降，そのデバイスコンテキストに対するグラフィックス描画を，指定した論理パレットを通じて行なうことができる．

パレットマネージャは，次のようにして各プログラムの論理パレットをシステムパレットに割り付ける．まずパレットマネージャは，その時点でアクティブなウィンドウの論理パレットをシステムパレットに直接マップする．

こうして，まだシステムパレットに空きがあるときには，バックグラウンドのウィンドウの論理パレットを割り付ける．そして，システムパレットがいっぱいになったら，残りの論理パレットには，システムパレットの中からそれに近い色を合わせる．このようにパレットマネージャは，アクティブウィンドウの論理パレットを優先的に割り付けるようになっている．

表5 Ver.2.Xにおけるビットマップファイルフォーマット

オフセット	サイズ	対応する構造体のフィールド名	内容
0	1	—	2に固定
1	1	—	破棄可能ビットマップの場合は80H，それ以外は0
2	2	bmType	0に固定
4	2	bmWidth	ビットマップの幅（ピクセル単位）
6	2	bmHeight	ビットマップの高さ（ピクセル単位）
8	2	bmWidthBytes	ワードバウンダリ（2バイトバウンダリ）にした場合のビットマップの幅（バイト単位）
10	1	bmPlanes	カラープレーン数
11	1	bmBitsPixel	1ピクセル当たりのカラービット数
12	2	—	0に固定
14	2	—	0に固定
16	2	bmBits	ビットイメージ（イメージのサイズに従って，データが繰り返し格納される）

表6 DIBフォーマット

オフセット	サイズ	対応する構造体のフィールド名	内容
0	2	bfType	ファイルタイプ．ビットマップの場合は“BM”
2	4	bfSize	ファイルのサイズ
6	2	bfReserved1	予約領域（現状では0に固定）
8	2	bfReserved2	予約領域（現状では0に固定）
10	4	bfOffBits	ファイルの先頭から数えたビットマップイメージへのオフセット
14	4	biSize	biSizeから，biClrImportantまでのサイズ
18	4	biWidth	ビットマップの幅（ピクセル単位）
22	4	biHeight	ビットマップの高さ（ピクセル単位）
26	2	biPlanes	カラープレーン数（DIBでは常に1）
28	2	biBitCount	1ピクセル当たりのビット数（1，4，8，24のいずれか）
30	4	biCompression	ビットマップの圧縮アルゴリズムの種類
34	4	biSizeImage	ビットイメージのサイズ（バイト単位）
38	4	biXPelsPerMeter	1メートル当たりの解像度（x方向）
42	4	biYPelsPerMeter	1メートル当たりの解像度（y方向）
46	4	biClrUsed	ビットマップが使用する色数
50	4	biClrImportant	ビットマップを表示するのに必要な色数
51	1	rgbBlue	B (Blue) の強度 ┐この部分は，カラーテーブルのエントリ数だけ繰り返される
52	1	rgbGreen	G (Green) の強度
53	1	rgbRed	R (Red) の強度
54	1	rgbReserved	予約領域（現状では0に固定）┘
55	—	—	ビットマップイメージ本体

*12 実際には，ビットマップファイルをロードする以外にも，プログラムでビットマップを作成することもできる．BITMAP構造体は，プログラムからビットマップを作成するときに使用する．詳細は，参考文献を参照のこと．

*13 たとえば，RGBという3プレーンでVRAMが構成されている場合には，bmPlanesが3，bmBitsPixelが1になる．

*14 同一のデータが連続する場合には，データを1つにして，連続する数をその直前に付けるという圧縮法．

*15 出力デバイスを仮想化するために，Windowsがシステム内部に保持しているデータ構造体．

Win3のユーザーインターフェイス

ここでは，プログラマの視点から見たWin3のユーザーインターフェイスについて述べる

ご存じのように，Win3のGUIは非常に洗練されたものになっている．このWin3のGUIを考えるうえで，ボタンやメニューなど，Win3が提供する部品が果たす役割は大きい．ここでは，こうした部品の中から，Win3で新たに追加されたものや，大幅に機能強化がなされたものをピックアップして紹介しよう．

コントロール

コントロール（Control）とは，ボタンやスクロールバーなど，主にユーザーからの入力を受け付けるための部品である．コントロールは，実際には子ウィンドウになっている．読者の多くも，ダイアログボックスにこれらのコントロールが表示されているところをご覧になったことがあるだろう．

もちろん，Windowsがあらかじめ定義しているコントロール以外にも，プログラマが独自に子ウィンドウを定義することが可能である．しかしこの場合には，キーボードのサポートなどをすべてプログラム自身で行なわなければならない．GUIで重要なのは，一貫した操作体系をユーザーに提供することにある．こうした意味でも，独自のコントロールを定義することは極力避け，Windowsがあらかじめ定義しているものを利用すべきだろう．すべてのアプリケーションプログラマが同じコントロールを使ってプログラムを作成すれば，複数のアプリケーション間でも操作方法が一貫したものとなる．

コントロールが子ウィンドウであることは今述べた．コントロールは，通常のウィンドウとまったく同様に作成され，ユーザーの入力情報（ボタンの押下など）をメッセージによってプログラムに通知する（あるいは，プログラムからコントロールに対してメッセージを送ることもできる）．

コントロールとしては，ボタン，スタティック（スタティックな文字列など），スクロールバー，チェックボックス，リストボックス，エディットコントロール，コンボボックスがある．ここでは，これらのうちリストボックス，エディットコントロール，コンボボックスを取り上げる．これらは，Win3で機能が強化されたものだ．特にコンボボックスは，Win3で新たにサポートされたコントロールである．そしてコントロールの最後としては，オーナードローボタンを紹介する．

■リストボックス

Win3のリストボックスでは，マルチカラムがサポートされた（**画面1**）．マルチカラムを使えば，ユーザーはカーソルを4方向に移動することができるようになる．これは，Ver.2.XのMS-DOS Executiveウィンドウで使われていた形式とよく似ている．カラムの幅は，プログラムで自由に設定できる．またこのため，リストボックスでは垂直スクロールバーに加えて，水平スクロールバーも使えるようになった．

複数の項目を選択できるリストボックスのスタイルは従来からあったが，これはトグル式で項目を選択するものであり，使いやすいとはいい難かった．Win3では，特定のキーを押しながらマウスでクリックすることにより，複数の項目を選択できる新しいスタイルが追加された．マウスをドラッグしながら複数の項目を選択することもできる．この選択方法は，ファイルマネージャでも用いられている．

Win3のリストボックスでは，文字列だけでなく，グラフィックスもリスト表示できるようになった．これは，オーナードローリストボックスと呼ばれるものである．

■エディットコントロール

エディットコントロールでは，入力する文字を大文字または小文字に固定できるようになった．また，パスワードやプライベートなテキストを入力するためのスタイルが追加されている．たとえばパスワード入力では，ユーザーが入力した文字は画面には表示されず，アスタリスクが表示される（アスタリスク以外の文字に変更することもできる）．従来は，ウィンドウのサブクラス化などを行ない，キーボードから入力された文字をプログラムで変換しなければならなかった．

■コンボボックス

コンボボックスは，リストボックスとエディットコントロールを組み合わせたものである．実際，コーディングの方法もそれぞれとほとんど同様である．

コンボボックスは，そのスタイルによって3種類に分けられる．1つは入力フィールドとリストボックスが常に表示されているタイプである．入力したテキストがリストボックス中に存在すれば，その項目がリストボックスの先頭に移動する．また，カーソルキーかマウスを使ってリストボックスから項目を選択することができる．

2つ目は入力フィールドの横にあるボタンによってリストボックスを開くタイプである（**画面2**）．リストボックスが表示されていない場合でも，カーソルキーを使って入力フィールド内にリストボックスの項目を選択できる．

最後は，入力フィールドが変更できない（スタティックになっている）タイプである．このタイプでは，リストボックス内の項目だけを選択できる．

コンボボックスを使えば，従来は必ず画面に表示する必要があったリストボックスを，必要なときにだけ表示させることができる．Win3のコントロールパネルを見ても分かるように，これによって効率よく画面設計を行なうことが可能になった．

■オーナードローボタン

オーナードローボタンは，機能的には通常のボタンとなんら変わりはない．ただしオーナードローボタンを使えば，ボタン上にプログラム独自のグラフィックスを描画できる．ボタンを描画する必要があるときには，Windowsがそのボタンを持つウィンドウに対してメッセージを送る．このメッセージにはボタンのID*16やボタンの座標などが含まれているので，プログラムではこれらの情報をもとにボタンを再描画すればよい．

メニュー

メニューは，一貫したGUIを構成するための重要な要素の1つである．たいていのWindowsアプリケーションはメニューバーを持っている．そして通常，メニューバーの項目が選択されると，ポップアップメニューが表示される．これは，ドロップダウンメニューもしくはサブメニューとも呼ばれる．

メニュー項目の定義は，通常はリソースファイル中で行なう．ただし必要ならば，プログラムからダイナミックにメニューを作成したり，メニュー項目を変更することも可能だ．

■階層化メニュー

Win3では，ポップアップメニュー内の項目から，さらにポップアップメニューを表示させることができる．これは，階層化メニューと呼ばれる(**画面3**)．ポップアップメニューの階層化を行なうのは非常に容易である．このためには，単にリソースファイル中のメニュー項目の定義をネストさせればよい．プログラムを変更する必要はいっさいない．

これまでのポップアップメニューを平面的なメニューと考えるなら，階層化メニューは立体的なメニューだといえる．階層化メニューを使えばすっきりと系統だったメニューを構成することが可能である．

■フローティング・ポップアップメニュー

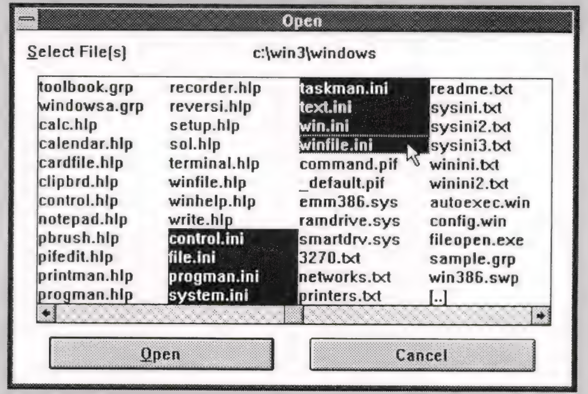
フローティング・ポップアップメニューを使えば，ウィンドウの任意の位置にポップアップメニューを表示させることができる．ただし通常は，マウス位置に表示するのが一般的だろう．頻繁にポップアップメニューの表示，消去を繰り返すようなプログラムでは有効だろう．

■オーナードローメニュー

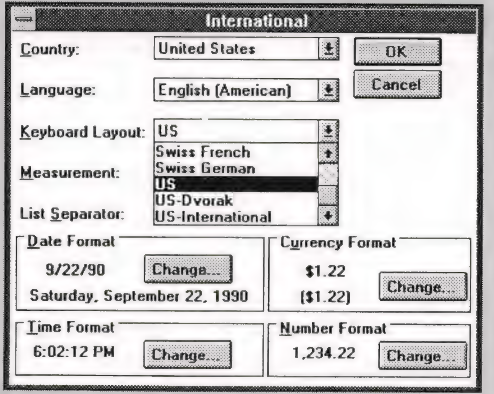
オーナードローメニューでは，リストボックスやオーナードローボタンと同様，メニュー項目としてグラフィックスを使用することができる．Ver.2.Xでもビットマップをメニューに表示することはできたが，オーナードローメニューでは，GDI関数を使って，よりフレキシブルにグラフィックスが描画できる．メニュー項目に対する描画のタイミング，および描画領域は，メッセージによってプログラムに通知される．

またWin3では，メニューを選択したときに表示されるチェックマークを，任意のビットマップを使って定義することもできる．　（遠藤　孝信）

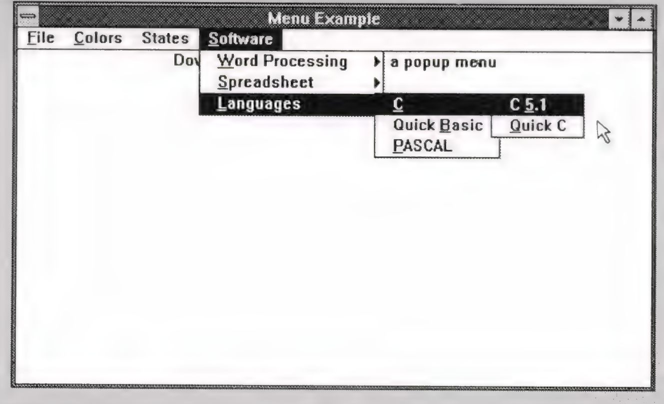
画面1　マルチカラムを使用したリストボックス



画面2　コンボボックス



画面3　階層化メニュー



＊16 ボタンの識別番号．

SDKが提供する開発ツール群

ここでは，Win3のSDKが提供する主要ツールの概要を述べ，その実力を探ってみよう

CVW (CodeView for Windows)

CVWに限らず，Win3のSDKに付属するデバッガを使用する場合には，セカンダリモニタまたはシリアル端末（デバッガによって異なる）が必須である．プライマリモニタにはWindows画面（デバッグ対象のウィンドウを含む），セカンダリモニタ（またはシリアル端末）にデバッガ画面を表示する．こうしてデバッグ対象の画面と，デバッガ用の画面を切り離すことで，実行画面に影響を与えることなく，デバッグを行なえるようになっている．

CVWはWindows対応のCodeViewデバッガである．CVWの利用には，セカンダリモニタが必要である（シリアル端末には対応していない）．

CVWは，CodeView形式のシンボル情報（EXEファイルの一部に格納される）を利用して，ソースレベルデバッグを実現する．他の２つのデバッガと異なり，CVWはWindows対応アプリケーションであり，Windows環境で起動することが可能だ．

CVWはMS-DOS用のCodeViewが備えている機能をもとに，Windowsアプリケーションのデバッグ用に以下のような新機能を備えている．

1.DLLのデバッグ機能
WindowsのDLLをデバッグできる．
2.複数プログラムの同時デバッグ
機能は多少制限されるが，DDEなどを使用したプログラムを効率よくデバッグできる．
3.メッセージに対するブレークポイント
指定した関数に特定のメッセージが送られた場合にデバッガに制御を移せる．
4.ヒープの利用状況の表示
Windowsのグローバルヒープ，ローカルヒープ中のオブジェクトリストを表示できる．

CVWは，Standard／386Enhancedモード専用のデバッガであり，Realモード環境下のプログラムはデバッグできない．Realモード環境のデバッグには，次に紹介するSYMDEBを使用する必要がある．

SYMDEB (Symbolic Debugger)

CVWと異なり，SYMDEBはDOSプログラムである．Windowsプログラムをデバッグする場合，SYMDEBのコマンドラインでWIN.COM（Windowsの起動プログラム）を指定し，Windowsカーネルごとデバッグ対象とすることになる．SYMDEBはCodeView形式のシンボル情報を扱えないため，リンカによってMAPファイルを作成し，それをSYMDEB用のSYMファイルに変換するという手間が必要になる．このためSDKには，Windowsカーネル用のシンボルファイルが用意されている．

SYMDEBは，セカンダリモニタとシリアル端末のどちらで使うことも可能だ．キーボードとマウスからの入力は，SYMDEBとWindows間で適宜振り分けられる．

SYMDEBの問題点は，コンベンショナルメモリにデバッガ本体が常駐してしまうことだ．このため，デバッグ時と通常時とでは，実行環境が大幅に異なったものになってしまう（SYMDEBは，EMSなどには対応していない）．

SYMDEBは，Realモードに対応した唯一のデバッガである．しかしその使い勝手は，あまりよいものとはいえない．CVWなどと比べると，見劣りするのはやむをえないだろう．プログラムの開発環境を見ても，Realモードが重視されていないのは明らかなようだ．

WDEB386 (386 Debugger)

WDEB386は，80386（DX／SX）／486（以下ではまとめて“386”と表わす）の32bitアーキテクチャに対応したデバッガである．8086／88や286などのシステムでは利用できない．ユーザーインターフェイスはSYMDEBに近く，ソースレベルデバッグができない点を考慮すると，３つのデバッガ中では一番プリミティブなデバッガといえる．

WDEB386を使用するメリットは，これが386に密着したデバッガであることだ．Win3は，386Enhancedモードで386特有の機能（ページングなど）を使用している．またすでに述べたように，386Enhancedモードでは SDK 付属の WINMEM32.DLLを利用して，32bitモードのプログラムを作成できる．こうしたプログラムのデバッグでは，32bit環境に対応したデバッガが必須になる．WDEB386は，Windows よりも下位，つまりWin3とハードウェアの間に入って処理を進める．このため，TSS*17やWindowsのメモリ管理に対応したコマンドが用意されており，よりハードウェアサイドに立ったデバッグが行なえる．

WDEB386を使用するには，シリアル端末が必須である（セカンダリモニタはサポートされない）．またWDEB386はStandardモード／386Enhancedモード専用のデバッガであり，Realモード環境のプログラムはデバッグできない．

プロファイラ (Profiler)

Win3のSDKでは，プログラムの最適化を支援するツールがいくつか追加されている．プロファイラ（PROF.COM）はその１つである．

プログラムを高速化するには，実行頻

度の高いコードを集中的にチューンアップするのが効率がよい．もちろん，プログラムの開発中もある程度の予想はするものだが，Windowsプログラムはユーザー入力に起因するものばかりでなく，システムから送られるメッセージも処理しなければならない．実際にプログラムを作成してみると，意外な部分で時間をとっていたというのはよくあることだ．

プロファイラは，ハードウェアのインターバル割り込みを利用して一定時間ごとに割り込みをかけ，その瞬間にプログラムのどの部分が実行されているかをカウントしていく．プログラム終了までに十分なサンプリングが行なえれば，プログラムの各部分の処理時間がほぼ確認できる．

SDKに付属するプロファイラは，Windowsプログラムのみならず，DOSプログラムの実行時間も計測できる．また，プロファイリングの開始／中断／再開／終了などをプログラムから制御できるので，プログラムの一部分だけを集中的に調査することも可能だ．なお，386Enhancedモードのプロファイリングは，PROF.COMではなく，WindowsのデバイスドライバVPROD.386によって実現される．

PROF.COMは，測定結果をバイナリ形式のファイルに格納する．これを表示するには，SHOWHITS.EXEを利用する．ShowHitsは，シンボルファイル(～.SYM)を参照し，セグメントごと，および各ブロック（関数ごと，といった単位）の実行時間の占有率を表示する．プログラムの高速化を図る場合には，貴重な情報を提供してくれるツールといえるだろう．

スワップチューナ (Swap Tuner)

Windowsプログラムでは，セグメント間コールのオーバーヘッドが非常に高い．また，Windowsではフリーメモリが不足すると，セグメント間コールのたびにセグメントのスワップや破棄のためにプログラムの実行速度が異常に低下する，“スラッシング”という現象が起きやすい．このような理由からWindowsプログラムは，セグメント間コールが極力少なくなるように作成する必要がある．

セグメント間コール／リターンと，それに伴うセグメントのスワップ／破棄を記録し，プログラム改良の参考資料を作成するツールがSwap Tunerである．Swap Tunerのデータをもとにモジュールを再構

＊17　Task State Segment. 80×86のプロテクトモードで使用するデータ構造の1つ．ハードウェアによってタスクを管理するためのもの．

改善された Win3のデバッグ環境

Windowsのプログラムをデバッグするとき，皆さんはどうされていたでしょう？ SYMDEBなどを用いてデバッグ情報をシリアル端末に出力したり，メッセージボックスを表示してメッセージの流れを考えながら，ソースリストと格闘したのではないでしょうか．もっと有効なデバッガがあったらどんなに楽かと考えたプログラマも多いと思います．Win3ではこの点にも改良が加えられています．ここではその一例として，Win3用のデバッガであるCVWに焦点を当て，実際の使用感を述べます．

写真に示したのは，IBMのPS/2に8514／Aディスプレイアダプタと8514ディスプレイを追加したシステムです．デバッグ用の画面は左側のディスプレイに，Win3アプリケーションの画面は右側のディスプレイに表示されています．従来のCodeViewでは，スクリーンスワップのオプションによってCodeView画面とアプリケーション画面を交互に切り替えてデバッグしていましたが，これでは作業がうっとうしいだけでなく，切り替えのタイミングによっては予期できないような事態になることがありました．CVWでは，2つのディスプレイを用いることにより，プログラムの内部を見ながら，実際のアプリケーション画面を観察できます．さらに，メッセージにブレークポイントを設定できる機能を用いることにより，問題箇所の発見が容易に行なえるでしょう．ただし，CVWの画面とWin3の画面との間での制御の移行には常に注意が必要です．

具体的なデバッグ作業の様子をここではお目にかけられませんが，このCVWはWin3のアプリケーションを開発するのはもちろんのこと，Windowsのプログラムを勉強するうえでも有効なツールとなることでしょう．しかし，CVWから提示される情報が多いだけに，十分に使いこなすまでにはそれなりの経験が必要です．これまでprintfのリダイレクションを使ったデバッグに慣れた方は，趣味に合わないと感じるかもしれません．まずは自ら体験されることをお勧めします．（佐貫　俊幸）

成すれば，実行時のセグメント間コールを削減することが可能になる．

Swap Tunerを使用する場合には，デバッグカーネル*18（SDKに含まれる）を使用しなければならない．また，プロファイラと異なり，Swap TunerはWindowsプログラム専用の解析ツールであり，DOSプログラムを扱うことはできない．

Helpコンパイラ (Help Compiler)

強力なヘルプ機能は，Win3のユーザーインターフェイスを支える特徴の1つになっている．このヘルプファイルの作成を支援するツールが，Helpコンパイラである．ヘルプコンパイラは，所定の規則に従って記述されたヘルプのソーステキストを，WinHelp（Win3のオンラインヘルプ・ビューワ）対応のヘルプファイルに変換する．

しかし，膨大な情報をオンラインヘルプにまとめることは，決して容易なことではない．Win3のオンラインヘルプでは，ビットマップを随所に配置できるばかりでなく，インデックスやハイパーテキスト風の参照が可能になっている．このためヘルプソースを作成するには，項目間の参照関係なども考慮しなければならない．また，ヘルプコンパイラの言語仕様は非常に大規模なものであるため，習得には一般のプログラミング言語と同程度の時間を割く必要があるかもしれない．

とはいえ，ヘルプソースで指定したMS-Wordなどの修飾指定の大部分がそのまま反映されるなど，便利な機能も多い．たとえばフォント指定や文字サイズ，あるいはクリップボード経由で取り込んだビットマップ（WinWordの場合）などは，そのままオンラインヘルプに表示可能である．

WinWordにおけるヘルプのソーステキストと，その表示結果を図4に示す（左上がソース，右下が表示結果）．このようにヘルプ中にボタンを配置することも可能である．（梅原　系）

*18　デバッグカーネルには，シンボル情報が残っており，このほかにもAPIコール時のパラメータをチェックする機能などがある．

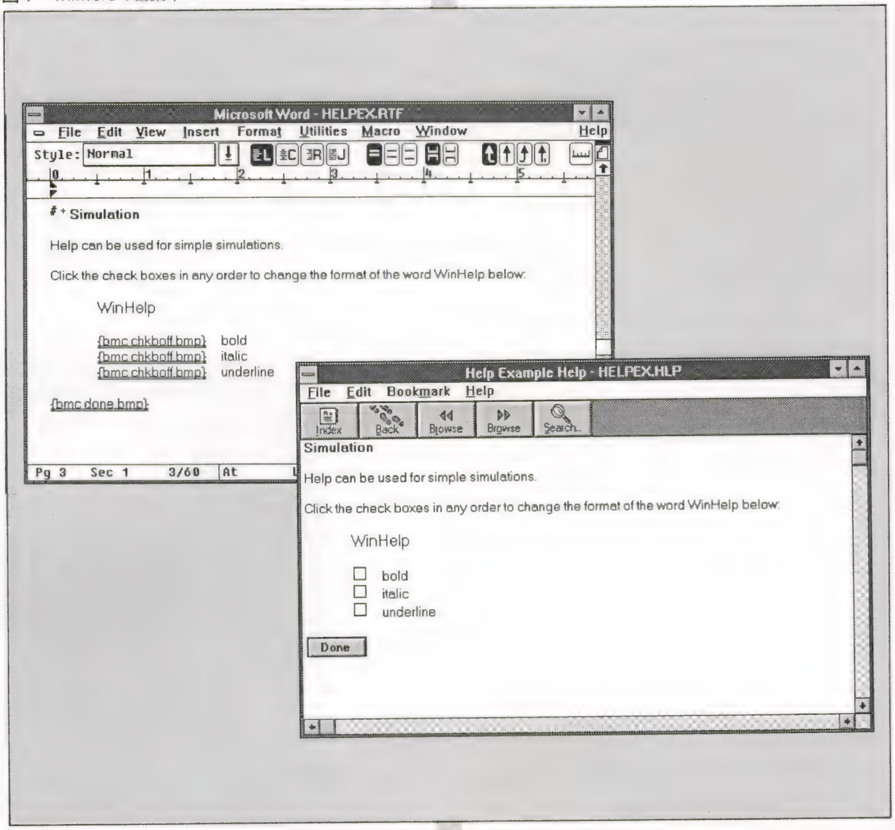
まとめ

開発ツールが大幅に強化されるなど，Windowsのプログラミング環境は整備されつつある．ドキュメンテーションはやや不足ぎみの感があるが，具体的なプログラミングについては，Microsoft Pressから今月（'90年10月）発刊が予定されている『Programming Windows Second Edition』（Charles Petzold著）に詳しく解説されるだろう．

Win3がプロテクトモードに対応されたことにより，デバッグ環境が著しく改善された点は見逃せない．従来は，メモリの制限から，デバッガを使用しづらいという問題があった．

Win3 SDKの日本語化も着々と進行しているものと思われるが，デバッガ用のセカンダリモニタの扱いは微妙だろう．IBM-PCでは，ディスプレイボードがメインボードとは完全に独立しているため，同時に2つのディスプレイボードを使用することは容易だ．日本語版のSDKにおいても，快適なデバッグ環境が提供されることを期待したい．（編集部）

図4　WinWordで編集中のヘルプソースとその表示結果



参考文献

●Microsoft Windows Guide to Programming, Microsoft Press社発行
●Microsoft Windows Programmer's Reference, Microsoft Press社発行
●Microsoft Windows Programming Tools, Microsoft Press 社発行
●"An Introduction to Microsoft Windows Version 3.0 : A Developer's ViewPoint" Michael Geary : Microsoft Systems Journal July1990, Vol.5 No.4, Microsoft Press社発行
●プログラミングWINDOWS Vol.1, Vol.2 Charles Petzold著，アスキー発行
●80286ハンドブック　大貫広幸，田中恵介，蓑原隆　共著，アスキー発行
●"HMA：リアルモードにおける64Kバイトの拡張領域" Chip Anderson：Microsoft Systems Journal日本語版 August 1989 No.2，アスキー発行

TEST LAB.

テスト ラボ

5月のビジネスショウで参考出品された，CPUに80486を採用したPC-H98がついに発売となった．価格，性能ともにワークステーションの領域に達したこのマシンを，はたしてどのような人がどのように使うのだろうか．今回は，このH98の性能を調べ，その存在意義について考えてみた．また，3年目でバージョンアップしたHyperCard 2.0についても，大幅に充実した開発環境を中心に紹介する．

PC-H98modelU100／60

高性能98はどこへ向かうのか

NWS-3460

小型，低価格のRISC NEWS

HyperCard 2.0

開発環境を大幅に充実

PC-H98model U100/60
高性能98はどこへ向かうのか

PC-98シリーズのトップモデルのPC-H98シリーズに新たに2機種の追加が行なわれた．CPUに80486を搭載したmodel100と80386のクロック周波数を25MHzとしたmodel60である．これにより，今までの1機種2モデルから，3機種10モデルと大幅な機種展開がなされたことになる．H98シリーズは，パソコンとはいえ，価格，性能ともにワークステーションに匹敵するものである．今回は，この限りなくワークステーションに近いパソコンの行く先をかい間見たいと思う．

80486を採用した最速98

model100は，PC-98シリーズで初めてCPUにクロック周波数25MHzの80486を搭載したマシンだ．メインメモリは，標準で7.5Mbytes，最大33.5Mbytesまで拡張可能である．内蔵ハードディスクの容量は100Mbytesで，ハードディスクを内蔵しないモデルは用意されていない．

model60は，CPUにクロック周波数25MHzの80386を採用している．メモリと内蔵ハードディスクの容量によって，3タイプに分かれている．002はメインメモリ1.5Mbytesでハードディスクは内蔵していない．040は同じくメインメモリ1.5Mbytesに，40Mbytesのハードディスクを内蔵している．100はメインメモリ5.5Mbytesに100Mbytesのハードディスクを内蔵している．それぞれ，メモリは最大27.5Mbytesまで拡張可能である．

model100，60ともに，フロッピーディスクドライブを2台内蔵しており，5インチドライブと3.5インチドライブのいずれかを選択することができる．3.5インチモデルには，モデル名に“U”が付きmodelU100というようになる．今まで，PC-98シリーズのハイレゾリューション(ハイレゾ）モデルには，フロッピーディスクドライブに3.5インチを採用したマシンは用意されていなかったが，今回発表されたH98からは，5インチモデル，3.5インチモデルをそれぞれ用意するとのことである．

価格は，model100が215万円，model60-002が83万5000円，040が99万5000円，100が150万円となっている．フロッピーディスクの5インチ，3.5インチモデルでの価格差はない．

表1に，現行機種のPC-H98model70と今回発表された製品のスペック一覧を挙げる．性能，価格的に，model100はmodel70の上位機種，model60はmodel70の下位機種にあたる．80万円強のマシンから200万円以上のマシンまで出揃ったことになり，NESAバス搭載のハイレゾマシンにも選択の幅が広がったことが分かる．

CPU以外の変更はなし

本体のサイズは，現行のmodel70と同じである．ただ，5インチと3.5インチのフロッピーディスクドライブで，本体を共用しているため，本体パネルの意匠が少々変更されている(**写真1**)．PC-9801RA21と比べると，設置面積は1.27倍ほどになる．

キーボードやマウス，バススロットなどの形状はmodel70と変更がない．各コネクタはPC-98シリーズと異なり，H98シリーズ独特のハーフピッチのものになっている．マウスのコネクタまで違うため，ほとんどのものを買い替える必要がある．マウス程度は同じにしてほしかった．

CPU以外のハードウェアスペックは，model70とほぼ同等である．ROMは64Kbytes(ハイレゾモード)または96Kbytes(ノーマルモード)，VRAMはテキスト16Kbytes，グラフィックス512Kbytes（ハイレゾモード）または256Kbytes(ノーマルモード）である（**写真2**）．

model60では，これまでのmodel70からの変更ではないが，64Kbytesのメモリキャッシュを内蔵している．速いCPUのマシンには，すでに必須である．

model70同様，グラフィックス処理にはE^2GCとAGDCを搭載している．どちらも従来のEGC，GDCのアッパーコンパチブルで，高速にグラフィックスを処理する．その能力の差については後述する．

バスにはNESA（New Extend Standard Architecture ）バスを採用している(**写真3**)．NESAバスは，model70で初めて採用された，従来からのPC-98シリ

ーズと互換性のある高速な32bitバスである．バスを使用する装置間の割り込み処理などの調整を行なうインテリジェントなものだ．NESAバスは，互換モードでもバスの調整機能は有効に働くので，RS-232Cの拡張ボードなど，今までならば1枚しか差すことができなかったボードを同時に使うことができる．

このNESAバスの採用により，H98シリーズでは初めて起動するときと新たに拡張ボードを差したときに，NESAバスのシステムセットアップが必要となった(**画面1**)．標準添付のリファレンスディスクにより，内蔵ハードディスクや拡張ボードなどの割り込み，DMAチャンネルやメモリアドレスの設定を行なう．内蔵ハードディスクは，自動的に適性値にセットアップされるのだが，NESAバスに対応していない拡張ボードでは，これらを入力する必要がある（NESAバス対応ボードは自動的にセットアップされる．また，日本電気純正ボードの場合は，リファレンスディスク内にこれらのデータが入っており，型番を選択するだけでよい)．通常，これらのデータは，マニュアルなどには記載されていないので，各ボードメーカーに問い合わせる必要がある．互換性を維持するためと，高度な調整機能によるためなのだが，割り込みチャンネルなどの意味を理解し設定するには，かなりの知識が必要だ．サードパーティ製のボードも純正ボードと同様に，データをリファレンスディスクに入れて，型番だけでセットアップできるようにすべきではないだろうか．

写真1　H98modelU 100の正面，側面，後面．5インチ，3.5インチモデルで本体を共用しているため，正面のデザインはmodel70と違っている．また，コネクタ類は，以前からのPC-98シリーズと違いハーフサイズのものとなっている．

互換性はメリットか

H98は，「速い98」である．H98と従来のPC-98シリーズとの違いは，NESAバスや高速なグラフィックスなど，内部的なハードウェアにある．ソフトウェア的には完全に互換性を持っているので，通常の使用ではそのスピード以外にとりたてて区別することはできない．グラフィックスの表示色の豊富さや，テキストアトリビュートが増えていることなどは，使ってみれば実感することもあるかもしれないが，現在のところ対応しているアプ

表1　H98シリーズの仕様

	model100／U100-100	model60／U60			model70	
		002	040	100	002	100
CPU	80486（クロック周波数25MHz）	80386（クロック周波数25MHz）			80386（クロック周波数33MHz）	
数値演算コプロセッサ	CPUに内蔵	オプション			オプション	
キャッシュメモリ	8 Kbytes（CPUに内蔵）	64Kbytes			64Kbytes	
メインメモリ	7.5Mbytes（最大33.5Mbytes）	1.5Mbytes		5.5Mbytes	1.5Mbytes	5.5Mbytes
		（最大27.5Mbytes）			（最大27.5Mbytes）	
フロッピーディスク	5インチ×2／3.5インチ×2				5インチ×2	
ハードディスク	100Mbytes	−	40Mbytes	100Mbytes	−	100Mbytes
ディスプレイ	ノーマルモード（640×400ドット） ハイレゾリューションモード（1120×750ドット） 1600万色中16色表示 （オプションで256色）					
サイズ	470（W）×345（D）×150（H）mm					
重量	15.0kg／14.0kg	13.5kg／12.5kg	15.0kg／14.0kg		13.5kg	15kg
消費電力	130W	100W	130W		100W	130W
価格	215万円	83万5000円	99万5000円	150万円	99万5000円	168万円

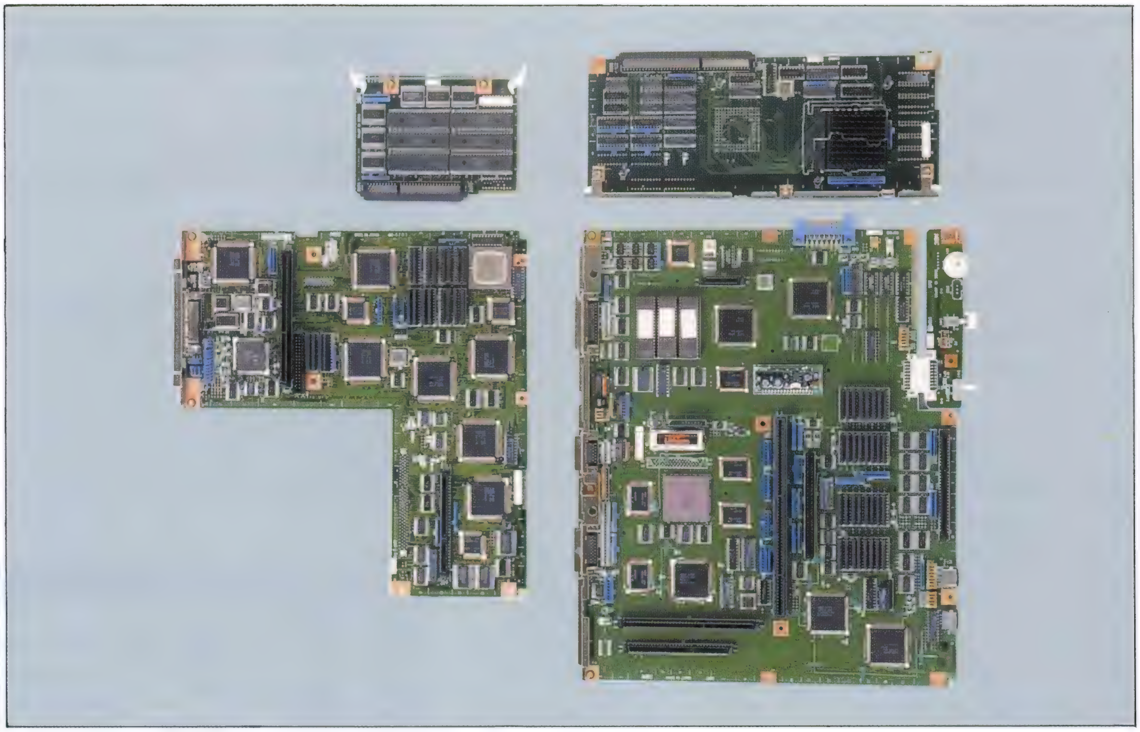


写真 2－1　model100の本体基板．巨大なマザーボードのほかに，CPUボード，グラフィックスボードなど４枚で構成されている．

リケーションがないため，実感することは難しい．

キーボードや，表示の雰囲気なども，RLなどのハイレゾマシンとほとんど変わるところがない．そして，その大きさは，なるほど98のハイレゾマシンらしいな，とまで感じさせるものとなっている．

もちろん，NESAバスを利用する拡張ハードウェアなど，H98に特化したものが普及すればH98のありがたみも大きくなることであろうが，今のところi860ボードといった高価なものしかなく，“宝の持ち腐れ”となっている．H98専用のソフトウェアも今のところ，MS-DOSや，MS-Windowsなどの，システムソフトウェアに限られてしまう．そのため，ソフトウェアもハードウェアも既存のPC-98シリーズ用のものを利用するしかない．ただ，従来のハード，ソフトが利用できるといっても，それにはいくつかの制限がある．

まず，ソフトウェアにはノーマルモードとハイレゾモードの問題がある．これは，H98に限った話ではなく，XL～RLといった一連のハイレゾマシンにも共通する問題であるが，これまでのPC-98シリーズのアプリケーションには，ハイレゾモードにきちんと対応しているものが少なく，通常ハイレゾで使用していても，アプリケーションによってはノーマルモードに切り替えなければならない．問題はノーマルモードとハイレゾモードでは，メモリマップなどがまったく異なるので，リセットを行なう必要があることだ．

ハイレゾマシンはハイレゾマシンとして使いたいのが人情である．従来との互換性のためだけにシステムを何度もリセットし，デバイスドライバなども別に用意するのはとても面倒である．ハイレゾ，ノーマルの問題は早急に何らかの解決をしてほしい．

ハードウェアには，前述のNESAバスのセットアップの問題がある．既存のボードを使用する場合においても，今までのPC-98シリーズで使えていたからといって，そのまま使用することはできず，NESAバスのセットアップが必要となる．そして，そのセットアップのためのデータは，各ボードメーカーに問い合わせなければならない．ハードウェアの知識を十分に持っていない人にとっては，この作業にかなりの努力を要するだろう．

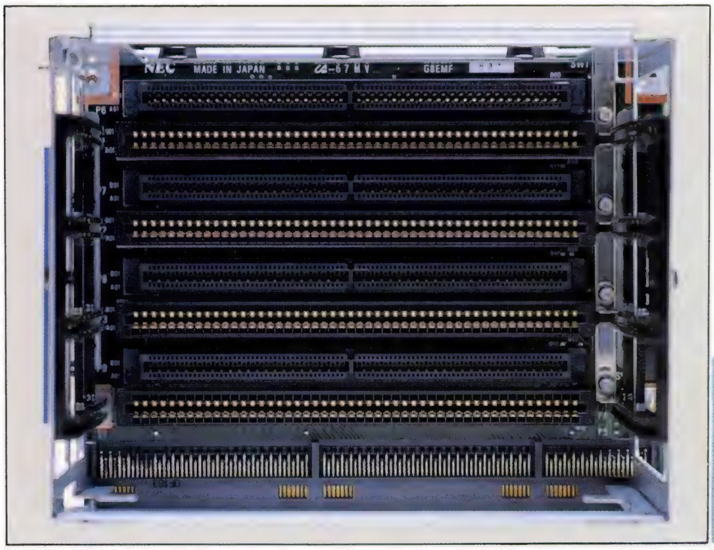


写真 3　NESAバスのスロット．上側のコネクタが32bitバスであり，下側のコネクタが従来と互換を持つ８／16bitバスである．

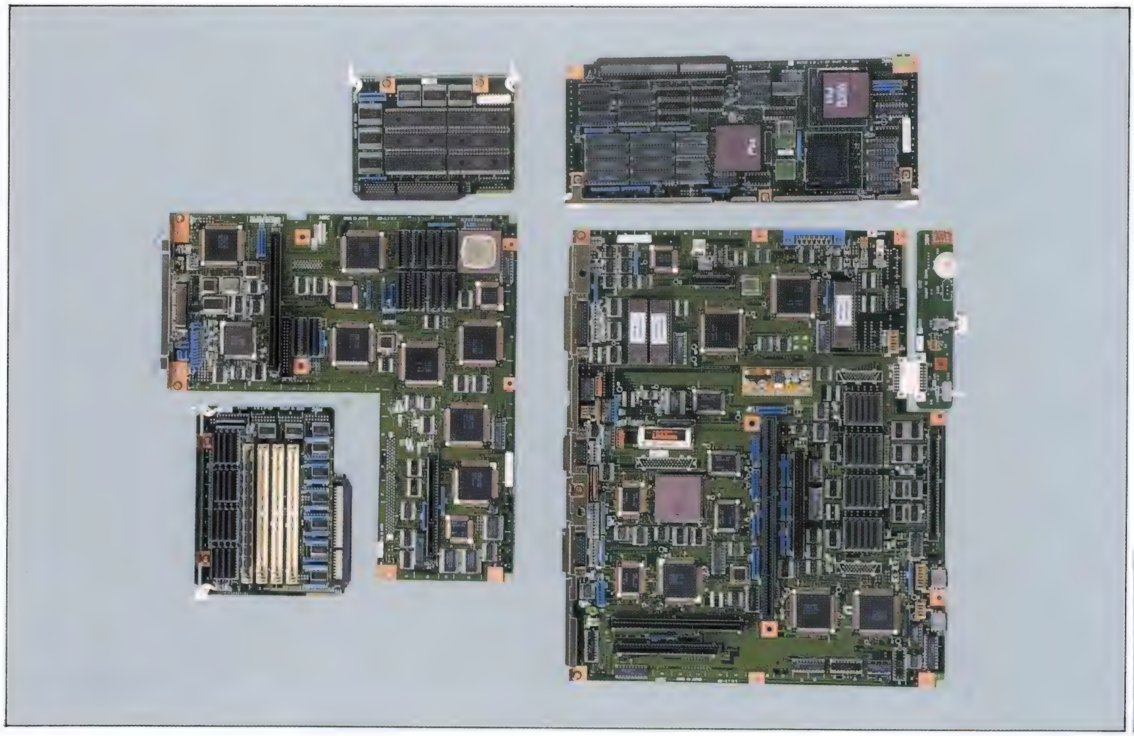


写真 2－2　model60の本体基板．model100のボード類にメモリボードが追加されている．CPUボードには，キャッシュメモリが搭載されていることが分かる．

キャッシュの効果を調べる

80486は，8086から80286へ，また80286から80386への移行に見られるような大幅な変更はない．ソフトウェア的には80386＋80387とほぼ同じである．

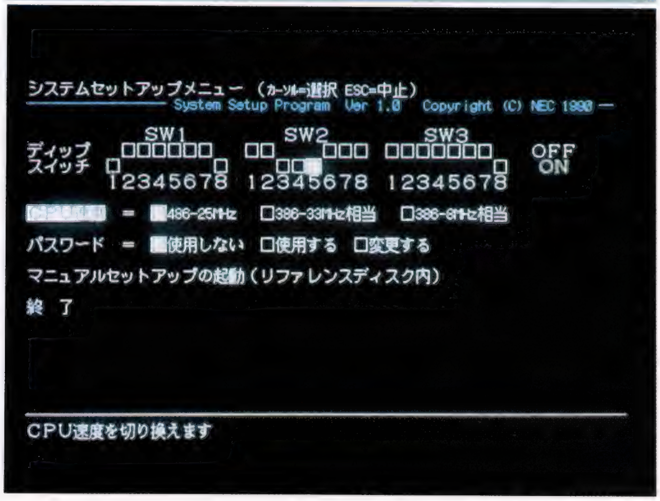
80486の特徴は80386と比べて高速化されていることにある．RISC技術の応用により，ほとんどの命令を1ないし3サイクルで実行する．また，インストラクションキャッシュを8 Kbytes持ち，メモリアクセスを減らしている．

インストラクションキャッシュとは，CPU内部に持つ命令先読み用のメモリである．キャッシュ内に次の命令があれば，遅い外部バスからメモリをアクセスしないので，命令のフェッチ時間を省略できる．その結果，高速な命令の実行が可能となる．

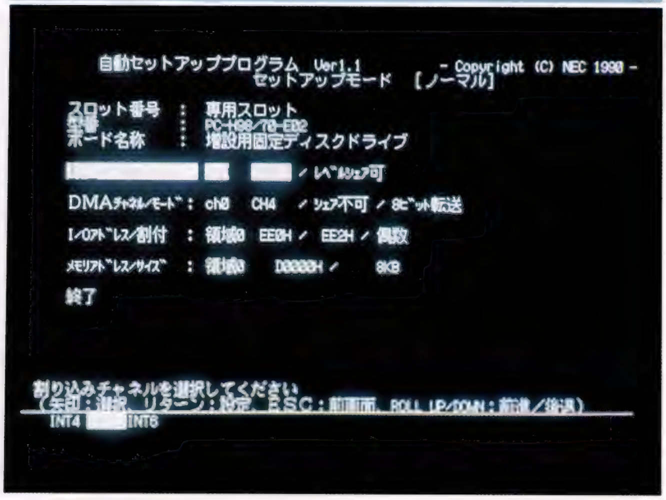
実際にどれくらい高速化されているかを確かめてみた．テストは，DhrystoneとWhetstoneで行なった．80386 (25MHz)のマシンとしてmodel60を，80486 (25 MHz)のマシンとしてmodel100を使用する．どちらのテストも十分短いので，80486のインストラクションキャッシュと80386の外部キャッシュにすべて収まるはずだ．

結果を**グラフ1**に示す．ご覧のように，Whetstoneでは約48倍，Dhrystoneでは約

画面 1－1　HELPキーを押しながらリセット(電源ON)をすると，SSP(SystemSetup Program) が起動し，画面のシステムセットアップメニューとなる．これにより，ディップスイッチは画面で簡単な説明を見ながら設定することができる．



画面 1－2　セットアップ画面で，“マニュアルセットアップの起動”を選択すると，付属のリファレンスディスクのバックアップディスクが要求される．画面は，内蔵ハードディスクのセットアップ画面である．設定には，割り込みチャンネルなどの知識が必要である．

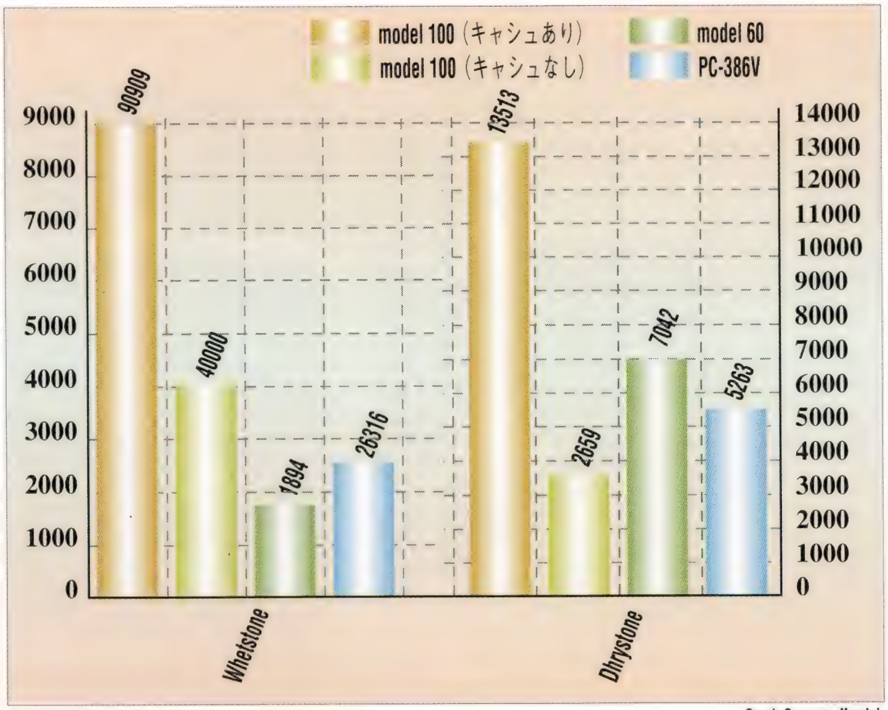


2倍の差が出た. model60には数値演算コプロセッサを搭載していないため, Whetstoneでは驚異的な差となっている. このうちどの程度が80486のインストラクションキャッシュの効果によるものなのかも, テストしてみた. テストのために, キャッシングを禁止して, 同じテストを実行した結果も示す.

キャッシングを禁止すると, キャッシュをフルに使用した場合と比べてWhetstoneでは約1/2, Dhrystoneでは約1/5程度の実行速度となった. model60と比べても, Dhrystoneでは半分以下のスピードである. また, クロック周波数20MHzの80386を搭載するPC-386V（外部キャッシュは持っていない. ただし, クロック周波数16MHzの80387搭載）と比較してもDhrystoneでは約1/2の速度である. 原因として, model60には外部キャッシュがあること, 80386もわずか数バイトではあるが, インストラクションキャッシュを持っていることが考えられる. Whetstoneでは, 数値演算コプロセッサをCPUに内蔵していることが効いて, Dhrystoneほどの速度の落ち込みがない. これらを見るとキャッシング機能と数値演算コプロセッサ内蔵の有用性がよく分かる.

ただ, このテストはどちらも極端な例である. 実際にアプリケーションを動かすときに, キャッシュがフルヒットするようなことは, まず考えられない. 逆に, すべてミスヒットすることも, 同様に考えられない. 同じクロック周波数の80386との実質的な速度差はキャッシュの効率や命令の実行速度から考えて, 一般的にいわれているように1.5〜2倍程度となるだろう.

グラフ1



Graph:Genmyo Kyodai

グラフィックス能力は1.5倍に

PC-98シリーズのMS-Windows／386 Ver.2.11には, PC-98版とPC-H98版の2種類があり, どちらもH98で動作する. 両方の版の違いは主に, H98で拡張されたグラフィックス機能——E²GCとAGDC——を使用しているかどうか, という点にある. はたして, これらの機能を使用することによって, どの程度パフォーマンスの向上が見られるだろうか, テストを行なった.

テストには, 他機種との比較も考慮して, Windows386ではなくWindows Ver.2.11を使用した.

また, このテストには, 編集部でXBenchをWindowsに移植したWinbenchを用いた. これは, スクロール, テキスト出力やブロック転送などの全体的なWindows上でのグラフィックスのパフォーマンスを調べるものである.

結果は, PC-98用のWindowsと, H98用のWindowsの比の形で示した. この差が, E²GCおよびAGDCを使用したときのパフォーマンスの差だと考えられる.

全体的に, 速度は向上しており, 全体で約1.5倍となった. 特に, 単純な直線や, 四辺形のパフォーマンスの差は非常に大きく, 3倍以上の速度となっている. CADなどのアプリケーションでは, このあたりが効いてくるだろう. ただし, ブロック転送（bitbltなど）の性能は, 変化がないようだ. E²GCやAGDCを使わずに, VRAMに直接転送することによって実現しているからと思われる.

ただ, 現状ではE²GCに対応したアプリケーションがないのが残念だ.

H98をどう使うか

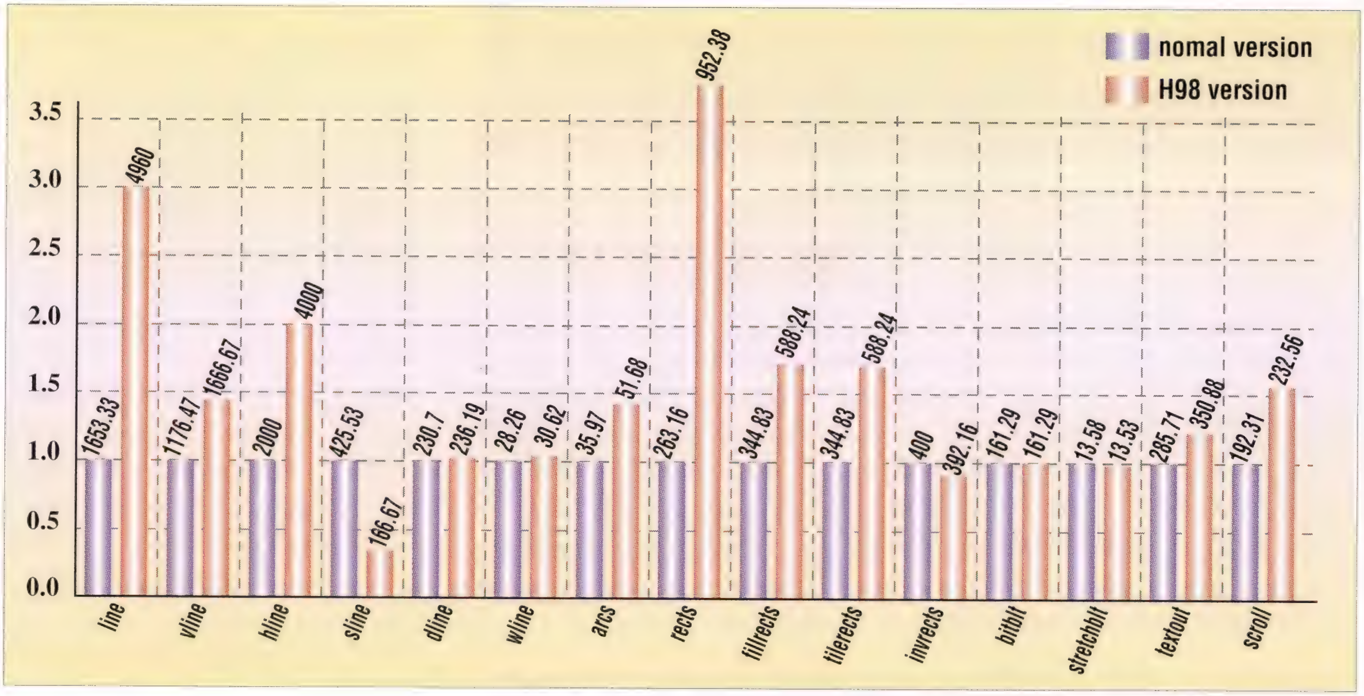
H98のメリットは, CPUのスピードもさることながら, NESAバスによるハードウェアの高速化とE²GCによるグラフィックスの高速化にある.

OS/2のプレゼンテーションマネージャや, Windowsなどのウィンドウシステムを使用する際に, 従来のマシンではグラフィックス処理のオーバーヘッドがネックになっていた. Winbenchの結果を見ても明らかなように, H98の高速なグラフィックスはこのオーバーヘッドを大幅に軽減する.

Macintoshのように, ウィンドウシステムの基本をファームウェアとして持つならば別だが, ウィンドウシステムの使用には高速な画面処理が必須である. また, ハイレゾリューションモードの画面は, 640×400ドットの狭い画面と違い, Windows等を使うのに十分な広さである.

矛盾するようだが, H98はファームウェアとしてウィンドウシステムを持っていないがゆえに, X Windowなどほかのウィンドウシステムへの対応も問題なく行なえる. H98用のX Windowサーバもいずれ発売されることだろう.

グラフ2



CADなどの高速なCPU，グラフィックス処理，およびハードウェアを必要とするアプリケーションを動かすマシンとしても，H98は適している．まだ，H98専用のCADソフトは発売されていないが，それほど遠くない将来に発売されることが期待される．

また，大容量のメモリ，ハードディスクが搭載，接続可能であり，それを活かしたネットワークサーバとしての利用も考えられる．現在のところパソコンのネットワークというのは一般的ではないものの，NetWareなどのネットワークOSの移植が進むことで，急速に普及することが考えられる．そのとき，PC-98シリーズのサーバとなるのは，やはりH98シリーズにほかならないだろう．

そして，UNIXワークステーションとしての利用も考えられる．ワークステーションとして使うなら，もともとH98を使う必要がない，という意見もあるかもしれないが，MS-DOSとのコンパチビリティを考えると，H98は優れたマルチOSマシンとなる．

PC-UXは，PC-98シリーズ用のUNIXである．このOSを使うとUNIXのアプリケーションばかりでなく，同時にPC-98シリーズのDOS用アプリケーションを動作させることができる．Sun386iのPC-ATモードのようなものだ．

PC-UXは近々バージョンアップしてSystemVR 4 コンパチになることがうわさされている．同時にH98用もリリースされるのだろう．そうすれば，H98はなかなか強力なワークステーションとなる．

互換性こそ重要だ

H98は，性能的に(そして価格的にも)ワークステーションにひけをとらないパソコンだ．そのパーソナルではないパソコンの登場の背景には，やはり従来のPC-98シリーズとの互換性があるのは間違いがない．

そのため，ワークステーションに比べて操作性や性能が犠牲になっていることも間違いのない事実だ．SPARCstation IPCが，H98model100とほぼ同じ価格でより高い性能であることがそれを証明している．

しかし，その互換性が重要であることも，また事実である．多くの企業で，これまでのパソコンでは，ますます重くなる業務に耐えきれなくなっている．だが，さらに高性能なマシンが必要だといって，すぐに今までの資産を捨て，ワークステーションに移行できるわけはない．また，移行してはたして将来性があるのかどうかも分からない．

H98は，そんなユーザーに向けられたマシンである．とりあえずは今までのソフトウェアが全部使え，今後発表されるであろう「重たい」ソフトウェアにも対応できる．さらには，高速なバスや拡張性を活かしてネットワーク環境などの高度な環境にも対応できるオールマイティなマシンである．

ここで最も重要なことは，これまでやってきたことをそのまま引き継げることにある．たとえ，ネットワーク環境などのより高度な環境に持ち込んでも，これまでの環境とも共存できるのである．

このように，企業ユーザーにとって，H98はとても安全で，将来性もあるマシンだといえる．

では，個人ユーザーにとってはどうであろうか．パソコン，オフコン，ワークステーションの性能が接近してきたことは，ある意味でチャンスといえる．パソコンの世界に飽き飽きしていたユーザーにとって，あと一歩で，別な世界に飛び込めるかもしれないのだ．ただ，安全とチャンスのどちらをとるかは，難しい問題ではある．　(上野和明)

NWS-3460

小型，低価格のRISC NEWS

「NWS-3400シリーズ」はMIPSのRISCプロセッサを搭載したソニーのワークステーションの新シリーズである．NEWSシリーズには，同じRISCプロセッサを搭載したNWS-3800シリーズがあるが，NWS-3400はその下位機種として，I／Oプロセッサを搭載しないことで，小型化，低価格化を実現したものである．現在有力なRISCプロセッサには，SUNのSPARCとMIPSのR3000がある．現在，これらのRISCプロセッサを用いたワークステーションが各社から発売されており，高性能ながら低価格であるため，今後のワークステーションの主流になっていくことが予想される．ソニーは今まで上位機種にはRISCプロセッサ，下位機種にはモトローラの68000系をCPUとして使用していたが，NWS-3400により68000系を使用したワークステーションと同価格帯でそれ以上の性能を得ることができるようになった．そのため，今後これが同社のワークステーションの中核となっていくものと思われる．今回は，この注目のNWS-3400シリーズを紹介する．

R3000を使用した低価格機

NWS-3400シリーズには415Mbytesハードディスク内蔵のNWS-3460とディスクレスのNWS-3410がある．

CPU，FPUにそれぞれクロック周波数20MHzのR3000，R3010を搭載し，整数演算で17MIPS，浮動小数点演算で2.3 MFLOPS（倍精度）の処理速度を達成している．

メインメモリは，標準で8 Mbytesを装備し，4 Mbytes拡張RAMキット（1 MbytesSIMM×4）を装着することにより，4 Mbytes単位で最大16Mbytesまで拡張することができる．クボタコンピュータ（MIPS）のRS3230が最大128Mbytesであることを考えると，最大16Mbytesというのは少ないように思われる．これは，増設メモリに4 MbytesのSIMMを採用していないためである．キャッシュメモリは，命令キャッシュとして64Kbytes，データキャッシュとして64Kbytesをそれぞれ内蔵している（**写真1**）．

インターフェイスは，Ethernet，SCSI，RS-232C×2，拡張フロッピーディスク，セントロニクスを備えている．また，マルチメディア対応のためオーディオインターフェイスを内蔵し，ヘッドホン端子とマイク，スピーカを内蔵したマイク／スピーカボックスが標準で付属している．音声データは，16／8bitのストレートPCMデータ，対数圧縮PCMデータの録音／再生，ADPCMの再生が可能である．

また，将来のFDDIネットワーク対応のためのFDDIネットワークボードを別売（価格195万円）で用意している．

ディスプレイは別売の専用のカラー，またはモノクロビットマップボードを専用スロットに装着して使用する．ディスプレイは，カラーで14／17インチ，モノクロで15／17インチのものを接続することができる．

本体のサイズは，ASICの採用で，非常にコンパクトにまとめられており，pop NEWSと同じ大きさとなっている（**写真2**）．とはいえ，最低限の拡張性は確保されており，1スロットではあるが拡張スロットを持っている．そのため，拡張ネットワークボードなどのボードが問題なく使用可能となっている．

NEWSシリーズの上位機種では，I／Oのスループットを上げるためI／Oプロセッサとして68030などのCPUを使っており，デュアルプロセッサを売りにしていた．NWS-3400シリーズでは低価格化ということもあり，I／OコントローラにCPUは使用していない．しかし，ソニー独自のASICによるDMAコントローラなどの採用で，DMA機能などは遜色のない程度に高速化されている．

価格は，NWS-3460が195万円，NWS-3410が125万円である．キーボード，マウスは別売で，セットで5万円となっている．また，カラービットマップボード（NWB-252）は35万円，モノクロビットマップボード（NWB-253）は10万円で，17インチカラーディスプレイ（NWP-519）は39万円，17インチモノクロディスプレ

イ（NWP-518）は12万5000円となっている。

OSはBSD4.3を元にしたRISC版

OSは4.3BSDをベースに，NFS Ver.3.2を搭載，日本語化したNEWS-OS3.91Rである．3.9からバグフィックスなどが行なわれ，3.91にマイナーバージョンアップされている．現行のRISC系の3.9とは完全に互換性があり，CISC系のNEWS-OSとはソース互換となっている．ちなみに，RISC系のNEWS-OSは年内にグラフィカルユーザーインターフェイスのmotifを標準添付したRelease 4 にバージョンアップされる予定である．

最近では，UNIXにあまり詳しくない人でも管理ができるように，ユーザー登録などの各種環境を簡単に設定できるセットアップコマンドを用意しているワークステーションが多い．しかし，NEWS-OSでは依然として設定に必要なファイルを書き換え，起動し直すという方法である．ただ，この点はRelease 4 でセットアップコマンドを採用し，改善されるとのことなので，どの程度簡単になるのか期待したいところである．

写真 1　NWS-3460の正面，側面，後面．本体サイズはPC-9801ESより小さいくらいだ．

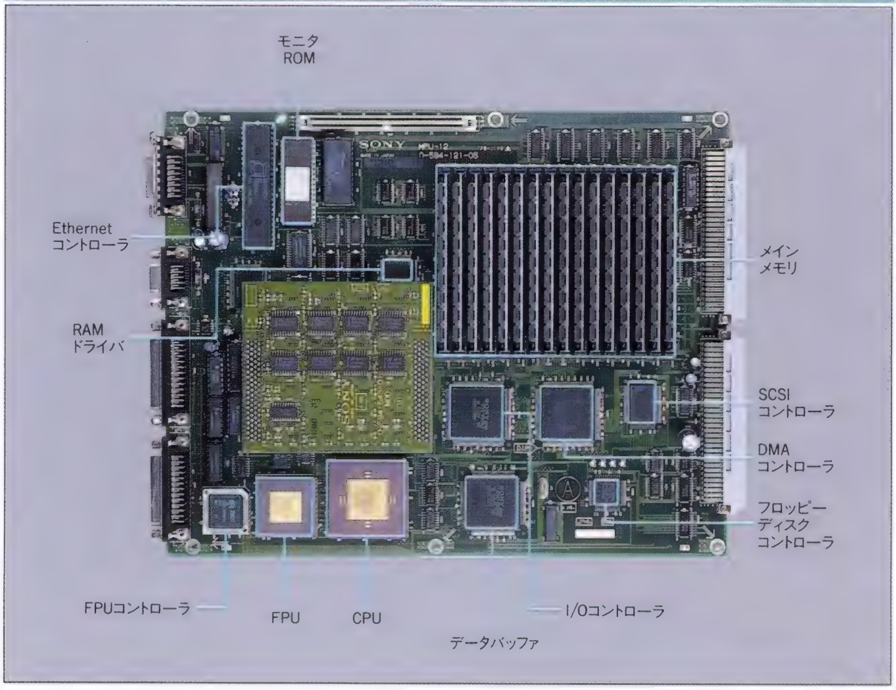
NEWS-OSでは，オンラインマニュアルが日本語化されている．これは，UNIXの使用される範囲が広がり，より多くの人がワークステーションを使用し，管理するようになることを考えると重要なことである．UNIXマシンでオンラインマニュアルといえば英文が普通であり，ここまで徹底した日本語化を行なったのはNEWSシリーズが初めてではないだろうか．

ウィンドウシステムには業界標準のX Window System Ver.11（X11）のRelease 3 が採用されている（**画面 1**）．X11のRelease 4 は，NEWS-OS Release 4 でサポートを行なうとのことである．motifのサポートは，Ver.3.91Rではなされていないため，ユーザーが独自にソースを入手してインストールするしかないのが残念だ．現状のユーザーのためにも，このバージョンでのmotifのサポートを行なってほしい．

NEWS-OSのX11に付属するコマンドには，標準的なX11のリリーステープに含まれるコマンド以外に，NEWS-OS独自のものが付属している．これは，NEWS-OSでX11を採用した当時，日本語化のためにjtermという端末エミュレータコマンドと，sjxというかな漢字変換システムを付属したためだ．今回のバージョンでは，端末エミュレータにはmtermというコマンドを用意し，jtermとhtermはそれにリンクされる形となっている．また，かな漢字変換システムは，以前からのsj2のほかに，sjxというX11上で別のウィンドウを開きそこからかな漢字変換を行なえるサーバと，サーバとの通信ライブラリが用意されている．

X11のリリーステープには，現在標準ともいえるktermという端末エミュレータコマンドとWnnというかな漢字変換システ

写真 2　メインボード．ASICの採用により，非常にコンパクトにまとめられている．



ムが入っているのだが，これらはX11のリリーステープを入手すれば，問題なくインストールできるだろう．

コンパイラは RISCompilerを採用

RISCプロセッサはその処理速度をパイプライン処理によって向上させており，プログラム実行時にパイプラインの流れを乱さないようにコンパイラが命令をスケジュールしなければならない．そのため，マシンの性能がコンパイラに大きく依存している．

MIPSのRISCプロセッサはコンパイラと同時に設計されており，その能力を最大限発揮できるのはMIPSのコンパイラなのは言うまでもない．そのため，MIPSのRISCプロセッサをワークステーションに使用するほとんどのメーカーがMIPSの最適化コンパイラ（RISCompiler）を採用している．RISCompilerは，アセンブラ，FORTRAN，C，PASCAL，COBOL，PL／I，およびADAをフロントエンドとし，それらの生成するucodeという共通の中間言語を用いて最適化を行なうものだ．

RISC NEWSシリーズでもRISCompilerを採用しているが，標準添付されるのはCコンパイラのみである．FORTRAN，PASCALは別売となっている．ちなみに，Cコンパイラのバージョンは，RS3230のRISC／os 4.50のものがVer.2.11，NWS-3400のものがVer.2.0であった．

RISCompilerは，RISC／os上で動作することを前提として作られており，オブジェクトファイルフォーマットとして，SystemVのCOFFを拡張したものを採用している．NWS-3400でも同様に，BSDフォーマットではなくこのCOFFを拡張したフォーマットを採用している．また，インクルードファイルは，コンパイラとライブラリ依存のものがMIPS，オブジェクトファイルフォーマットに関係したものがAT&TのものをMIPSが拡張したもの，システム関係のものがBSDのものを，それぞれ元にしている．Cコンパイラに標準のライブラリは，MIPSのものを元にしている．

画面1　X11の画面．中央のウィンドウは移植したNemacsである．ユーティリティにはISDN対応のもの（左上）も入っている．

Nemacsの移植を実施してみた

今回試用したマシンの構成は，メインメモリが8Mbytes，ディスクが415Mbytesで，カラービットマップボードに20インチのカラーディスプレイを付けたものである．

各種の環境を整える手始めに，現在のワークステーションの標準的な環境ともいえるNemacsの移植を行ない，CISC系のNEWS-OSとの違いや開発環境などについて調べてみた．

初めは，オプティマイズレベル2を指定してグローバルなオプティマイズを行なった．マシン定義ファイルは，MIPSのものがあったためこれを使用した．また，システム定義ファイルは，BSD4.3のものを使用した．しかし，ライブラリがMIPSのものを元にしているため，これではうまくいかなかった．そこで，マシン定義ファイルをBSD4.3のものを元に，MIPSの定義ファイルの中から，リンク時の定義を取り出して加えたものに書き換えメイクをやり直した．これで，Nemacsは問題なくコンパイルでき，動作させることができた．

この後に，グローバルなレジスタ割り当てを含むすべてのオプティマイズを行なうように，オプティマイズレベルを3に指定してNemacsを作り直すことを試みた．前にも述べたように，MIPSのコンパイラは，ucodeでオプティマイズを行なうので，このままではこのオプティマイズはできない．そこで，メイク時のコンパイラの出力を，オブジェクトファイルからucodeオブジェクトファイルとなるようにmakefileを書き換えた．

また，emacsは標準のCのスタートアップモジュールの前に，別のオブジェクトをリンクする必要があるため，コンパイルの中間処理であるオプティマイザの処理なども，makefileにいちいち記述してしまうことになった．これは，コンパイラに渡すオプションスイッチの設定で解決できるのかもしれないが，オンラインマニュアルなど調べた結果では不可能なようである．このコンパイラを使いこなすには，オプティマイズの過程などかなりの知識が必要である．なお，RS3230にバンドルされるemacsはオプティマイズレベル2でコンパイルするように設定されていた．

ちなみに，標準的なプログラムではmakefileの簡単な書き換えだけで，オプティマイズレベル3のコンパイルを行なうことができる．試しに，OPEN LOOKのウィンドウマネージャをコンパイルしてみたが何も問題なく，あっさりとコンパイルすることができた．emacsのような変わったことをしなければ，ほとんど大丈夫なようだ．

速度，価格にバランスのとれたマシン

	NWS-3400	RS3230
CPU	R3000（クロック周波数20MHz）	R3000（クロック周波数25MHz）
数値演算コプロセッサ	R3010（クロック周波数20MHz）	R3010（クロック周波数25MHz）
MIPS値	17MIPS	20MIPS
メインメモリ	8 Mbytes（最大16Mbytes）	8 Mbytes（最大128Mbytes）
フロッピーディスク	3.5インチ（1.44Mbytes）内蔵	3.5インチ（1.44Mbytes）内蔵可能
ハードディスク	415Mbytes（NWS-3460）	600Mbytes内蔵可能（最大6.6Gbytes）
ディスプレイ	1024×768ドット 1280×1024ドット	1280×1024ドット（カラー） 1152×900ドット（モノクロ）
標準インターフェイス	Ethernet, SCSI, RS-232C×2，オーディオ	Ethernet, SCSI, RS-232C×2
拡張スロット	1	ISAスロット（モノクロモデル）
標準ソフトウェア	NEWS-OS 3.91R	RISC／os 4.5
サイズ	355（W）×341（D）×110（H）mm	406（W）×406（D）×102（H）mm
重量	約10kg	17kg
消費電力	55W	220W
価格	125万円（NWS-3410） 195万円（NWS-3460）	199万円（カラー標準モデル，ディスプレイ付き）

表1　NWS-3400の仕様

NWS-3460は，ソニーが初めて発売したワークステーションであるNWS-821と並べて使用するとその処理能力の差にあらためて驚かされる．ワークステーションの進歩の速さを感じさせられる．

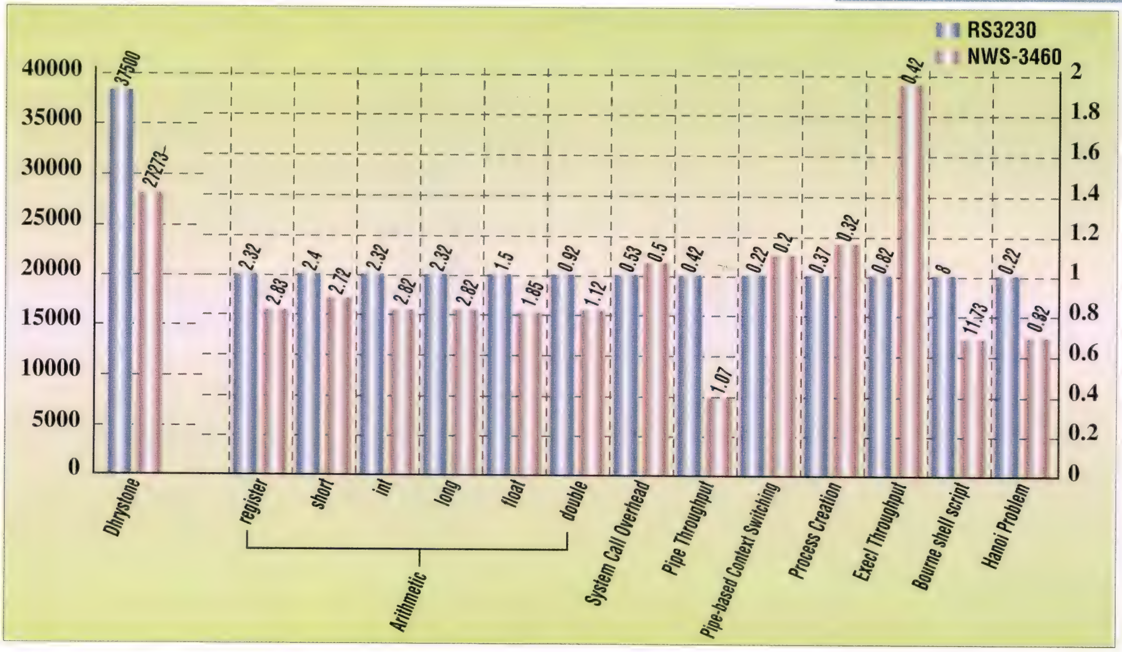
ベンチマークテストの結果は，この価格帯のマシンとしては非常に良い結果を出している(**グラフ1**)．しかし，RS3230と比較すると，ほとんど同価格でありながら，RS3230のクロック周波数が25MHzと少々高速であるため，RS3230のほうが多くのテストでさらに良い結果を出した．pipeやforkなどのシステムコールの性能を計るテストでは，NWS-3400のほうが良いものもあるが，これはOSの違いによるものである．ユーザータイムを多く使うテストでは，RS3230のクロック周波数を20MHzに換算してもRS3230のほうが勝っている．この結果は，ハードウェアの違いのほかに，コンパイラのバージョンによる影響が大きいと思われる．メモリの容量は，ベンチマークのプログラムがどれもそれほど大きくないため，あまり影響はなかった．実際に，実用的なNemacsなどのプログラムを使用したときも，NEWS-3400のほうがキャッシュメモリの容量が多いためか，その差は気になるほどではない．ただ，X11上で使用すると，たまにメモリのスワップが起こり動作が遅くなることがあったので，メモリはなるべく増設したほうがよいだろう．

SPARCstationIPCについては，まだベンチマークテストを実施していないが，公称MIPS値では，NEWS-3400のほうがわずかに勝っている．SPARCstationシリーズは，豊富なアプリケーション数を売りにしているが，RISC NEWSシリーズでも年内に約150本のアプリケーションを用意するとのことだ．

カラーモデルの標準構成の価格を，RS3230と比較すると，RS3230がメインメモリ8 Mbytes, 200Mbytesハードディスク，17インチカラーディスプレイで199万円，NWS-3460がメインメモリ8 Mbytes, 415Mbytesハードディスク，17インチカラーディスプレイ（カラービットマップボードを含む）で274万円となっており，少々高くなっている．ただ，ハードディスクの容量などを考慮すれば，R3000を搭載したワークステーションとしては最低価格帯に位置しているマシンの1台といえる．

（北原 直樹）

グラフ1　ベンチマークテストの結果



HyperCard 2.0
開発環境が大幅に充実

1987年夏のHyperCardの登場は，ユーザーにもソフトウェア業界にも，大きな衝撃を与えたことは間違いない．HyperCardが切り開いた新しいソフトウェアのスタイルは，Macintosh（以下Mac）以外の機種にもいくつもの類似ソフトを生みだした．そして，それは今日に至って，1つの新しいソフトウェアのジャンルとして認知されている．そのHyperCardの新バージョンが，6月に発表された．実に4年目にして，初のメジャーなバージョンアップとなる．HyperCardはこの3年間，細かなバージョンアップを繰り返してはきたが，大幅な機能改変はこれが初めてのことだ．本誌9月号でもレポートしているAsymetrix社の「ToolBook」など，他機種上の類似ソフトが台頭する中，本家HyperCardがどのような機能アップを図ったかを注目してみたい．

バージョンアップの概要

HyperCard 2.0のリリースは，現在のところ未定であり，この原稿の執筆時点ではリリースされていない．そのため，試用したのは英語版のベータバージョンである．正式リリース版とは画面などの細部が異なる可能性があることを，あらかじめお断りしておく．

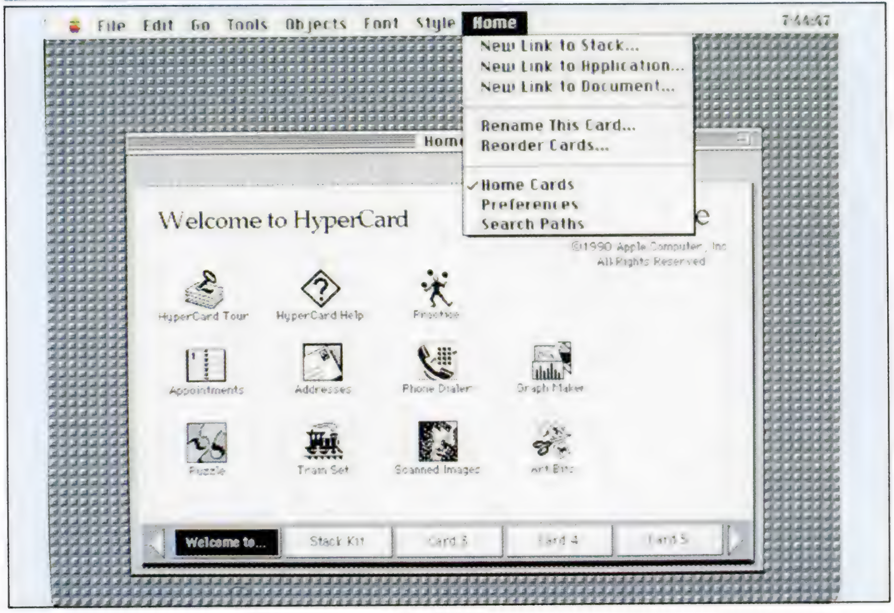
まず，2.0の1.xからのバージョンアップに伴う，主な機能の強化点は，

・マルチウィンドウで複数のスタックを同時にオープン可能
・スタックごとにカードのサイズが変更可能
・メニューの作成が可能
・スクリプトエディタがマルチウィンドウになり，複数のオブジェクトのスクリプトを同時に編集可能
・スクリプトデバッガを搭載
・テキストハンドリングの強化
・プリント機能の充実
・搭載スクリプト言語「HyperTalk」の強化

と，大雑把にざっと見渡しただけでも，これだけの項目を挙げることができる．細部になると，変更点はかなりの広範囲にわたる．また，機能の強化／変更に伴って，2.0のスタックは機能的には上位互換となったのだが，フォーマットが変更され，1.xまでのスタックとの互換性はなくなってしまった．しかし，2.0には，1.xのスタックを新しいフォーマットにコンバートする機能が用意されているので，実用上の問題はまずないだろう．

以降は，これらの変更点を踏まえながら，2.0の概要を解説していく．

画面1　一新されたデザインのHomeスタック．独自のメニューを装備して，使い勝手が向上している．



一新されたHomeスタック

具体的な変更点に触れる前に，まずHomeスタックの様子から見ていこう．

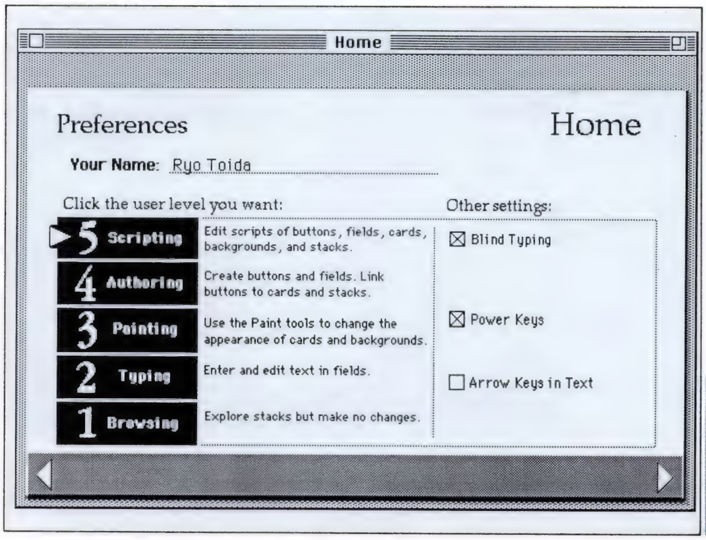
Homeスタックはデザインが一新された以外に，HyperCardのシェルとしての役割は，1.xからの変更はない（**画面1**）．ユーザーの利用形態に合わせて，5段階のユーザーレベル*1を設定できる点も同様だ（**画面2**）．

ただ，一見してこれまでと異なっていることが分かる点として，Homeメニューの存在がある．

従来のHomeスタックでは，ユーザーが新規に別のスタックを呼び出すボタンを作成する場合，まず新しいボタンを作成し，そのボタンと目的のスタックをリンクするという面倒な作業が必要であった．それが，2.0のHomeスタックでは，Homeメニューの「New Link to Stack...」コマンドを実行することで，ボタンの作成から目的のスタックの選択，作成したボタンの配置までをごく簡単に行なえるようになった．このHomeメニューは，2.0で追加されたスクリプトによるメニュー作成機能を利用して作られており，ユーザーインターフェイスの改善が図られている．

また，Homeスタックと並んでHyperCardの基本的なスタックであるオンライ

＊1　HyperCardでは，ユーザーのHyperCardの利用形態に合わせて，5段階のユーザーレベルを設定している．ユーザーレベルには，既存のスタックを利用するだけの「Browsing」から，HyperTalkでのプログラミングを行なう「Scripting」までがあり，レベルごとに利用できる機能が異なる．



画面 2　Homeスタックのpreferencesカード．利用形態や目的に応じて，5段階のユーザーレベルが選択できる．

ンヘルプも，「HyperCard Help」と「HyperTalk Reference」の2部構成となり，項目ごとのリファレンスも充実し洗練された内容のものが付属している（**画面 3**）．

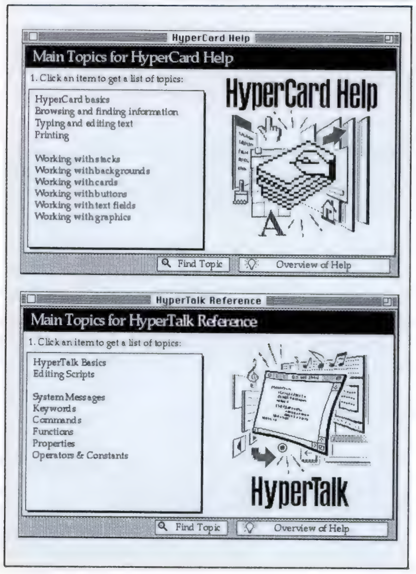
複数のスタックを同時にオープン

数ある2.0の強化点の中でも，やはり，カードの大きさが変更可能になったことと，同時に複数のスタックをオープンして取り扱えるようになったことの2点は，最大のポイントである（**画面 4**）．

従来の1.xでは，同時にアクセスできるのは1つのスタックのみで，カードのサイズはMacPlusの9インチモニタの大きさ（512×342ドット）に固定されていた．そのため，1画面（1枚のカード）に表示できる情報量が強く制限を受けることとなり，根強い不満の1つとなっていた．今回の強化により，この不満はとりあえず解消されたわけだ．

カードのサイズはスタックごとに，最小は64×64ドットから最大で1280×1280ドットまで，縦横1ドット単位に設定できる．サイズは，基本的にスタックの作成時に指定するのだが，使用中のスタックでも変更可能だ（**画面 5**）．残念ながら，同一スタック内では異なるサイズのカードを混在させることができない．しかし，これは複数のスタックを，同時にオープンできることで解決できる問題である．

画面 3　オンラインヘルプスタックは，HyperCard Help（上）とHyperTalk Reference（下）の2部構成となった．



同時にオープンできるスタックの数には，基本的な制限はなく，搭載している メモリ容量が許す限り，いくつものスタックを開いておくことができる．

新たなユーザーインターフェイスPalette

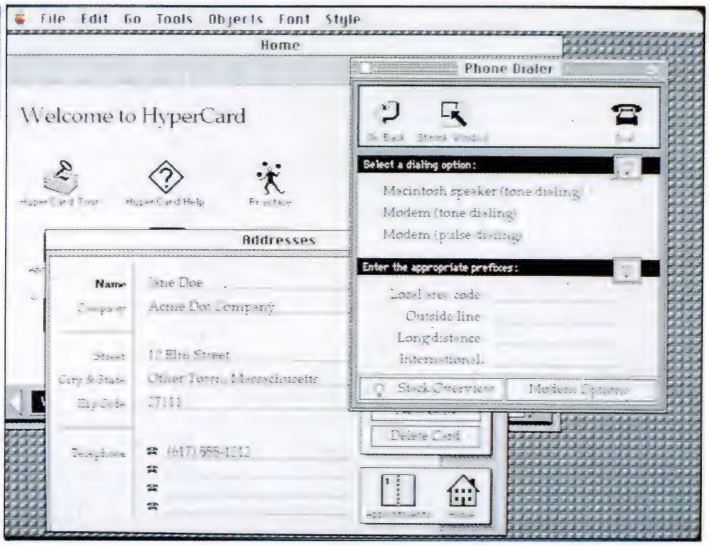
先の2つの変更点に関連して，追加された便利な機能が2つある．

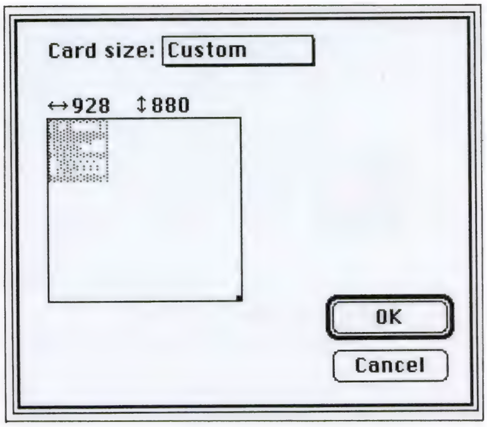
1つはScrollツールと呼ばれるものである．アクティブなスタックのウィンドウサイズをカード本来の大きさよりも縮小したとき，ウィンドウ内でカードをスクロールするためのツールだ（**画面 6**）．このツールは，MacIIの13インチモニタサイズで作成されたスタックを，MacPlusなどの小さなモニタで利用する場合などに必要な機能として用意された．

もう1つは「Palette」と呼ばれる新しいHyperCardのユーザーインターフェイスで，2.0には標準で「Navigator」と呼ばれるサンプルのPaletteが付属している．Navigatorは各種のアイコンをクリックすることで，カードを移動したり，Homeスタックに復帰したりと，GOメニューに用意されているコマンドと同等の働きをする（**画面 7**）．

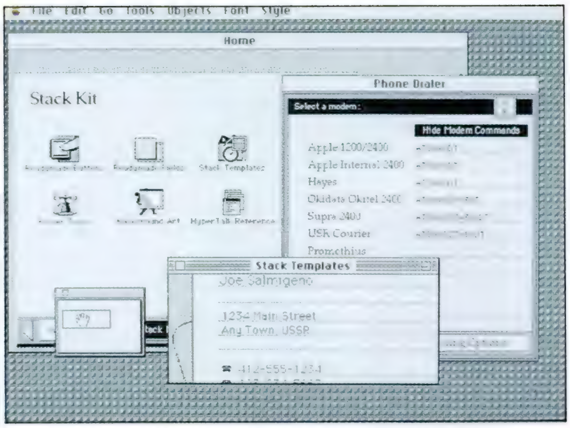
この2つの新しい機能は，外観上は従来からティアオフメニュー（tear-off menu）として用意されていたToolパレットやPatternパレットと同じタイプのウィンドウ*2で実現されているが，Palette

画面 4　複数のスタックを同時に参照することが可能になった．





画面 5 カードサイズを設定するダイアログ．現在のカードの大きさがグレイで表示され，枠線の右下端をマウスドラッグすることで縦横１ドット単位で指定できる．



画面 6 画面左側の小さなパネルウィンドウがScrollツール．パネルウィンドウ内の白い領域が，アクティブなスタックのカードの大きさに相当し，その内側の黒枠線がウィンドウの大きさ／表示位置を表わす．黒枠線をマウスドラッグすることで，ウィンドウの大きさ，表示位置を変更できる．

(Navigator) はユーザーが自由に作成できる点が異なる．サンプルのNavigator上のカードを移動するボタンのように，スタックには少なからず共通して利用されるボタンがある．そうしたボタンをPaletteにまとめることで，スタック開発の手間が省け，ユーザーの操作性も向上するというアイデアだ．

Paletteは，パレットウィンドウの絵柄（アイコンなど）になるPICTリソースと，パレットウィンドウの大きさやマウスクリックされた場合の振る舞いを定義したPLTEリソースの２つから構成される．PLTEリソースは，XCMD*3やXFCN*3のようにMacの実行コードを収めたものではなく，Paletteの大きさやクリックされた絵柄に対応する１ライン分のHyperTalkのスクリプトを収めた比較的簡単な構造のものだ．

Paletteの作成ツールは，今回試用したベータ版には付属していなかったが，正式版には用意されると思われるが，現在のところ未定である．いずれにせよ，それほど複雑なものではないので，いざとなればPLTEリソースの作成ツール（それこそスタックで）を自作することもできるだろう（**画面 8**）．

Paletteは同一のスタック内で，異なるサイズのカードを取り扱えない制限を補うものではないが，自分のオリジナルのスタックで，頻繁に使うボタン類を独立したパネルウィンドウとして用意できるのならば，スタック作成の自由度は飛躍的に高まるはずだ．

＊２ 一般にwindoidと呼ばれるタイプのウィンドウで，通常のドキュメントウィンドウよりも常に上側に表示される．多くのアプリケーションで，ツールアイコンの表示ウィンドウなどに利用されている．

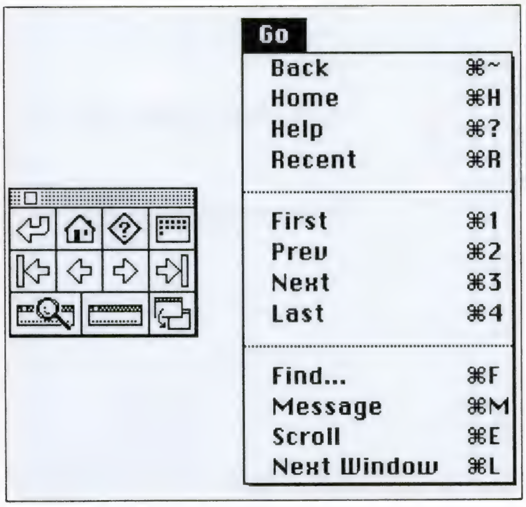
＊３ Hyper Cardでは，Hyper Talkで実現できない機能を，外部プログラムを利用して実現するためのインターフェイスを備えており，XCMDはHyperTalkのコマンド，XFCNはファンクションと，同じようにHyperTalkから利用できる．

一段と自動化が進んだボタン編集機能

HyperCardといえば，なんといってもボタンが主役だが，ボタンに関してもいくつかの変更が加えられている．

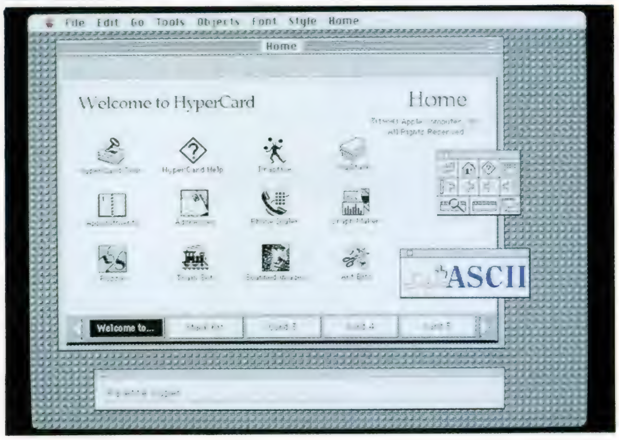
これまでも，ボタンとほかのカード（あるいはスタック）のリンクは，マウスでリンク先を指定することでボタンに，格納するスクリプトが自動的に作成できた．2.0では，それに加えてワイプなどのビジュアルエフェクトも，いちいちスクリプトエディタを呼び出すことなく，一覧表からエフェクトを選択するだけで作成できるようになった（**画面 9**）．つまり，視覚効果を伴ってカードを移動するボタンは，まったくスクリプトを書かなくても作成できるようになったのだ．

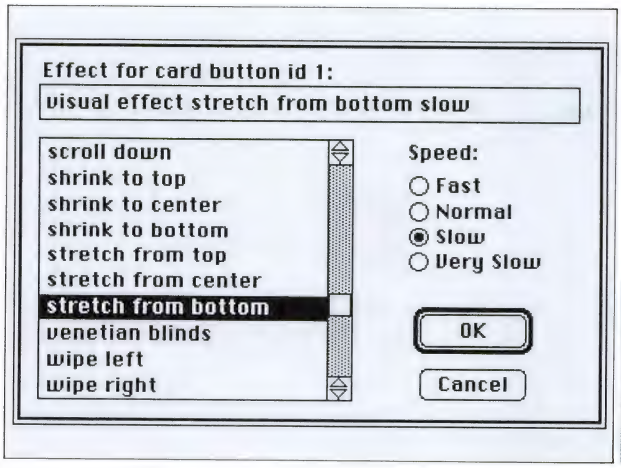
さらに，アイコンエディタも搭載されて，ボタンに付けるアイコンの編集もHyperCard内で行なうことが可能となっ



画面 7 PaletteのNavigatorは，カードをめくったり，Homeスタックに復帰するなど，GOメニューのコマンドと同等の機能を果たす．

画面 8 Paletteは，ユーザーが作成し自由にスタックにインストールして利用できる，2.0の新しいユーザーインターフェイスだ．





画面9　ボタンに格納するビジュアルエフェクトのスクリプトも，一覧表から選択するだけで作成できるようになった．ビジュアルエフェクトの数も，shrinkとstrechのバリエーションを加えて22種類と豊富だ．

た．アイコンエディタを起動するとアイコンエディタ専用のメニューが現われ，イメージの反転や回転などの編集が行なえる．画面中の任意の部分を，32×32ドットの範囲でピックアップし，アイコンエディタに転送する機能も備えられているので，絵の中にアイコン（ボタン）を埋め込むような場合に威力を発揮するだろう（**画面10**）．

また，もう一方の主役であるテキストフィールドも，1つのフィールド内に複数のフォント，サイズ，スタイルの混在が可能になり，StyledTextに対応した(**画面11**)．これまでは，フィールドの属性として設定したフォントでのみ表示していたのだが，ワープロなどと同じようにフィールド中の文字列を選択して，異なるフォントやスタイルをメニューから指定できるようになった．さらにテキストについては，HyperTalkもハンドリングを強化して，フィールド中の任意の文字列をボタンと同様に，ユーザーからのアクションを受信する器として利用できるようにもなっている．つまり文書中の1語を指定して，別のカードを参照するなど，よりハイパーテキストらしい取り扱いが可能になっている．

スクリプトエディタとデバッグ機能

HyperTalkによるスクリプティングのための環境は，大幅に改善されている．

まずスクリプトエディタがマルチウィンドウになり，複数のオブジェクトのスクリプトを同時に編集することが可能となった（**画面12**）．Cut，Copy，Pasteなどの基本的な編集コマンドに加え，検索・置換機能も備えて，通常のテキストエディタと同様な操作性を実現している．

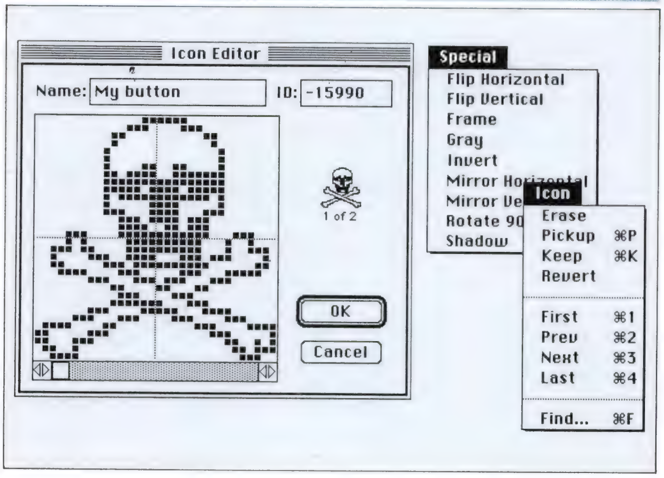
従来のスクリプトエディタは，縦方向のスクロールしかできず，しかもCut，Copyなどのコマンドはキーボードショートカットしか受け付けないという，お世辞にも使いやすいとはいえないものだった．それだけに，スクリプトエディタの改善を，長く待ちわびていたユーザーも少なくないはずだ．

このスクリプトエディタは，単にスクリプトの編集を行なう以外に，次に述べるデバッグ機能のチェックポイントの設定にも利用できる．チェックポイントは，他のデバッガでいうところのブレークポイントに相当するものだ．チェックポイントをスクリプトの任意の行に設定しておくと，そこでスクリプトの実行が一時中断され，デバッグ機能をアクティブにすることができる．

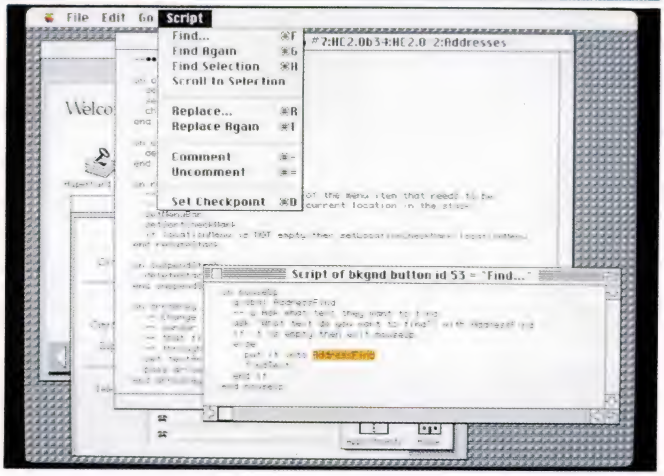
デバッグ機能が呼び出されると，各種のデバッグコマンドを実行するDebuggerメニューと，デバッグ対象のスクリプトが自動的に現われる．Debuggerメニューには，スクリプトの実行状況を監視しながら1行ずつ断続的に実行する「Step」コマンドや，連続して実行する「Trace」コマンドなど，ほかの言語処理系のソースレベルデバッガと同様な機能が備えられている（**画面13**）．

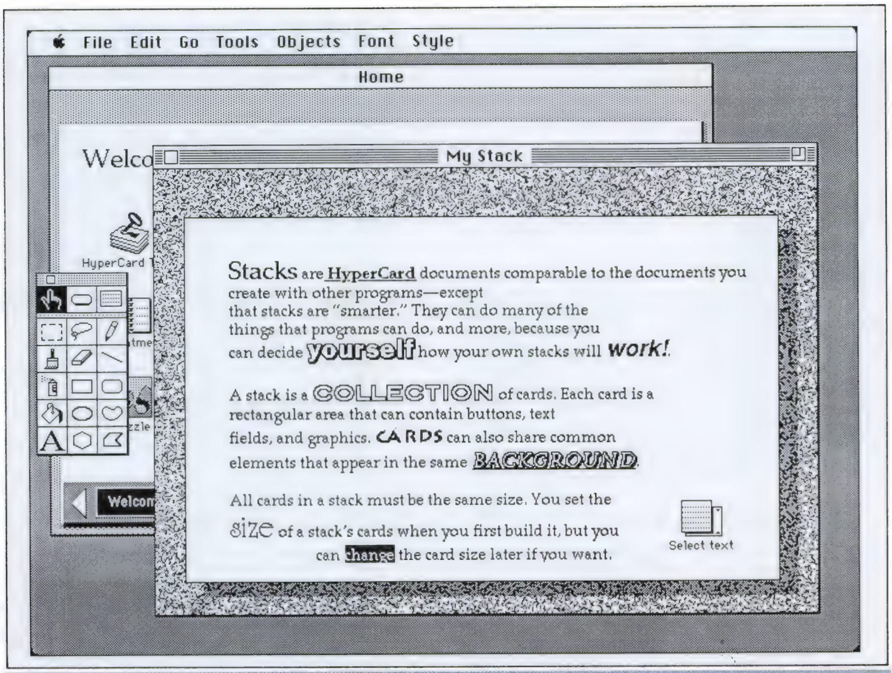
デバッグ機能では，単にスクリプトのトレース実行を行なうだけではなく，「Variable Watcher」と「Message

画面10　アイコンエディタのダイアログ．カレントのスタックに登録されているアイコンを切り替えながらの編集も可能．



画面11　テキストフィールドは，StyledTextに対応しワープロのような感覚でフォントやスタイルを指定できる．



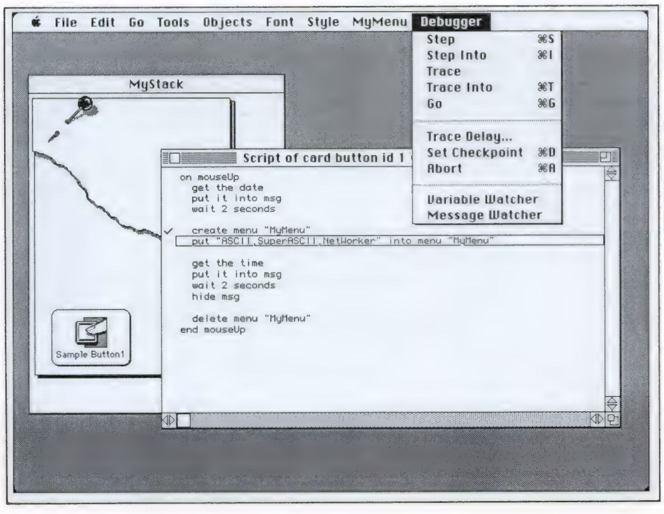


画面12 スクリプトエディタは，検索置換機能など，まともなテキストエディタの体裁を整えた．

Watcher」という，2つのユーティリティを併用して，より本格的なデバッグ作業が可能になっている．Variable Watcherでは，現在実行中のスクリプト(メッセージハンドラ)が利用するすべてのローカル／グローバル変数を表示し，その値を編集することができる(**画面14**)．Message Watcherは，システムメッセージや，スクリプトから発せられたメッセージを150個まで記録し，その内容を確認することができるものだ(**画面15**)．

デバッグ機能は，チェックポイントによるもの以外に，エラー発生時にも起動することができる．2.0のスクリプトのエラーには2種類あり，1つはスクリプトにミスタイプや文法的な誤りがある場合に発生する「Compile-time error」，もう1つは実行不可能な処理がある場合に発生する「Run-time error」だ．Compile-time errorの発生時には，スクリプトエディタを起動できるだけだが，より複雑なエラーのRun-time errorの発生時には，デバッガを利用することができるという2段構成をとっている．

ちなみに，スクリプトエディタやVariable Watcher，Message WatcherはXCMDでHyperCardに搭載されている．このため，たとえばサードパーティからより強力なスクリプトエディタがリリースされることも，十分に可能性があることだろう．



画面13 チェックポイントを設定してデバッグ機能を呼び出せる．Debuggerメニューの「Step Into」や「Trace Into」を利用すれば，他のオブジェクト内のスクリプトの実行まで追跡してみることができる．

インヘリタンスの構造の変化

先に2.0では，複数のスタックを同時にオープンできると述べたが，それにはもう1つ面白い方法が用意されている．スタックを画面に表示せずにオープンすることができるのだ．つまり，ユーザーが直接操作するためではなく，ライブラリファイルのように，そのスタックに収められているスクリプト（やXCMDなど）を利用することを目的とする方法だ．通常の方法でオープンしたスタック間では，メッセージの交換やスクリプトの共有ができないのだが，この方法によってHomeスタック以外のスタックを，複数のスタックからアクセス可能な共有ライブラリとして利用できるのだ．

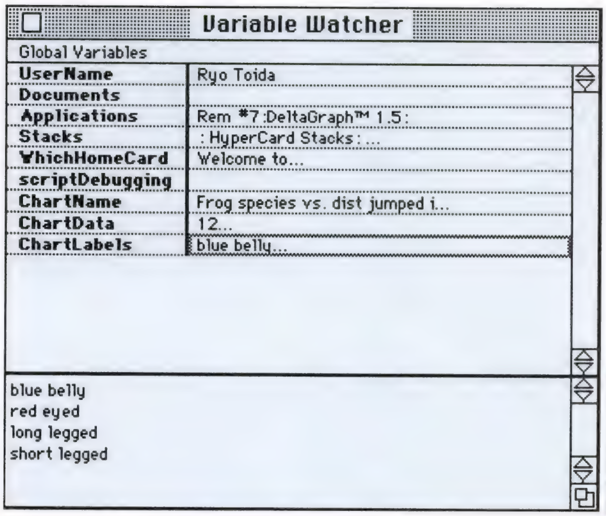
この方法には，拡張されたHyperTalkのコマンド「start using」と「stop using」を以下のように使用する．

start using stack StackName
——StackNameで指定したスタックの使用を開始する．

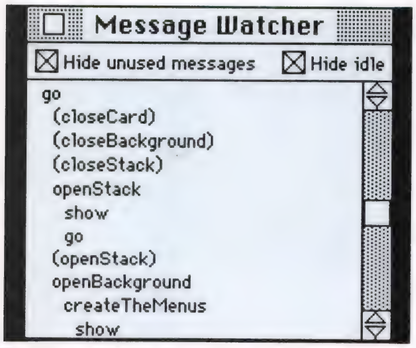
stop using stack StackName
——StackNameで指定したスタックの使用を終了する．

この「start using」コマンドを使って，同時に最大10個までのスタックを不可視の状態で使用できる．使用中のスタックは，「stacksInUse」というプロパティで確認可能だ．

従来のメッセージパス（スクリプトのインヘリタンス）は，ボタン（フィールド）から始まって，カード，バックグラウンド，スタック，Homeスタック，HyperCard本体へとつながっている．ところが「start using」コマンドでオープンした不可視のスタックは，従来のスタックとHomeスタックの中間に加えられた形になるのだ(図



画面14　Variable Watcherのウィンドウ．スクリプトのデバッグ中に，ローカル／グローバルの変数を表示し，その変数を任意に編集できる．



画面15　Message Watcherは，スクリプトの実行中に発せられる各種のメッセージを150個まで記録してくれる．記録されるメッセージには，スクリプト実行開始のトリガとなるmouseUpやkeyDownなどだけではなく，ハンドラの中から発せられたメッセージも含まれる．また発したものの使用されなかった（無視された）メッセージは括弧書きで表示する．オプションの指定で，そうしたメッセージの記録を抑制することもできる．

1）．そのため通常の状態でオープンしたスタックからは，不可視のスタック内のスクリプトはHomeスタックのスクリプトと同じ扱いとなる．

これは，上位オブジェクトのスクリプトを継承するHyperCardの「インヘリタンス」の構造を拡張する方法として，思えば単純だが効果的な対処の仕方だろう．2.0ではメニューやPaletteなど，ユーザーインターフェイス部分の拡張も施されている．そのため，同時に使用する可能性のある個々のスタックが，同じような機能のメニューやPaletteを持つことは，スタック開発の面でもメモリ消費の面でも非合理的だ．そうしたときに，不可視のスタックによる資源（メニュー，Paletteなど）の共有が生きてくる．

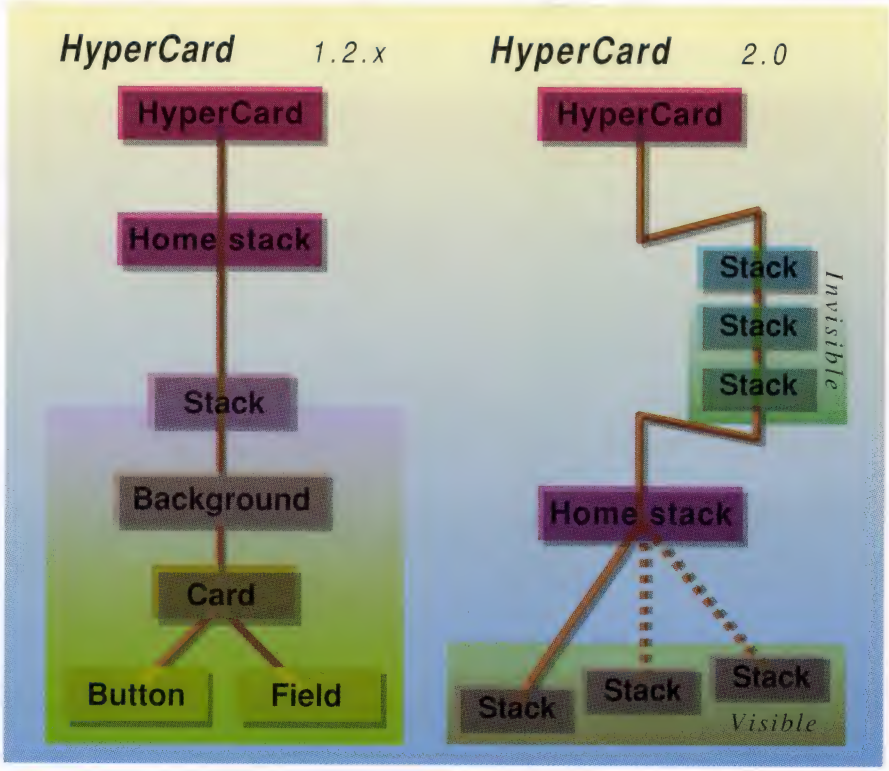
「start using」コマンドのほかにも，HyperTalkは非常に数多くの強化／拡張が加えられている．HyperTalkは，HyperCardのほとんどすべての機能をスクリプトで制御できることが特徴であるから，外観上変化した機能はすべてHyperTalkに反映されていると思って間違いない．さらに，外観の変化を伴わないHyperTalk自体の強化も数多くある点は見逃せない．

☆

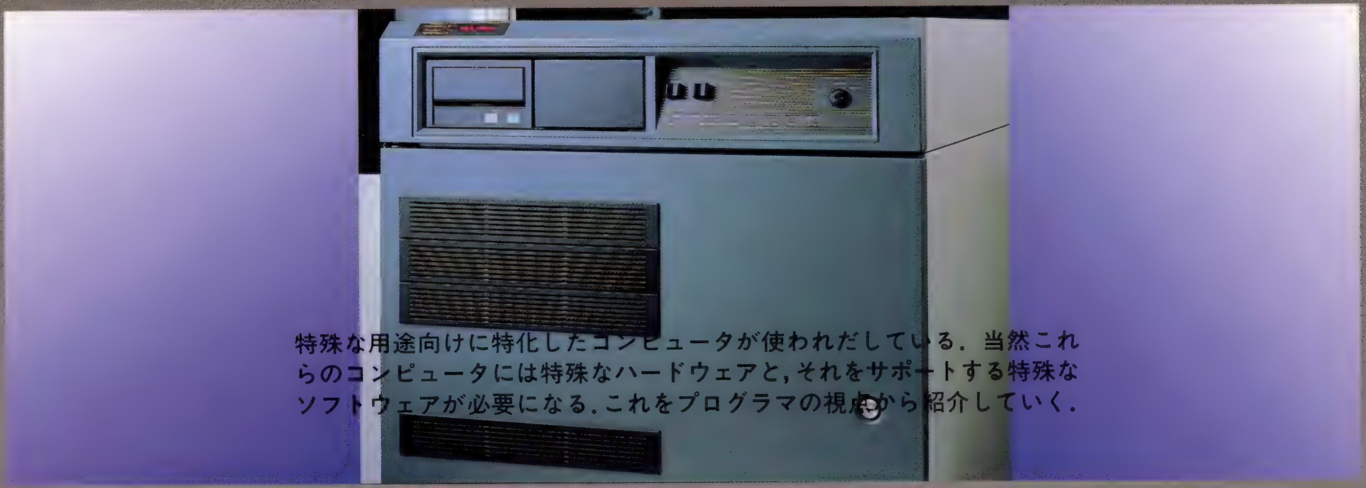
HyperCardは，今回のバージョンアップで当初の予想以上にパワフルに，格段に使いやすくなった．目立つところでは，開発環境の充実が興味深いが，かなり細部にわたって全面的な見直しがなされたのだろうという感想を抱いた．大幅な機能拡張に伴って，HyperCardの「真のエンドユーザーが気楽に向き合えるアプリケーションプラットホーム」という持ち味が少し薄れたかと予想していたのだが，それは間違いだった．逆に，1.xのHyperTalkでは不明瞭だった部分，あるいは扱いにくかった部分がしっかりとした分だけ，応用しやすくなった．ただ，カード上でドローグラフィックスが扱えないとか，カラーグラフィックスに対応していないなど，まだまだ不満な点があるのも事実である．　　（樋田　亮）

製品名	HyperCard 2.0
対応	Macintosh Plus以上
価格	未定
問い合わせ先	アップルコンピュータジャパン ☎：03-224-7000

図6　従来のHyperCardではのメッセージパス（メッセージの伝達経路：左）と，「start using」コマンドで拡張されたメッセージパス（右）．2.0ではスタックをスクリプトの共有ライブラリとして取り扱うことが可能になっている．



400MIPSを目指す並列コンピュータ
Multimax



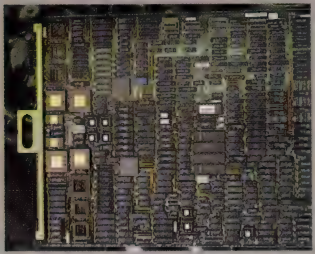
特殊な用途向けに特化したコンピュータが使われだしている．当然これらのコンピュータには特殊なハードウェアと，それをサポートする特殊なソフトウェアが必要になる．これをプログラマの視点から紹介していく．

'80年代の研究テーマであったオブジェクト指向は，'90年代には一般化し，'90年代の研究テーマは高並列システムであるといわれている．その理由は，OLTPやネットワーク環境の規模の拡大とともに，処理時間の短縮，信頼性と稼働性を向上させることが，社会のニーズとして顕在化してきたためだと考えられる．

並列化自体の研究は意外と古くから行なわれており，すでに1961年には，バロースのB5000というデュアルシンメトリ・プロセッサが紹介されている．そして，'70年代から'80年代において，並列化に関するコンピュータアーキテクチャ，プログラミング言語，アルゴリズム，応用分野などの研究が行なわれ，その結果VLIW，パイプライン処理，マルチプロセッサなどといった形で並列機構は現実化されてきた．数多くの研究の中からいくつかのアイデアは商品化されているが，今はまだ特殊な分野の問題解決に当てられているにすぎない．

写真1　Multimaxのシステムキャビネット（高さ1300mm×横幅710mm×奥行き710mm）．これには，主にプロセッサボードやメモリボードが収められている．

写真2　300シリーズに使用されているAPC．NS32332のほかに，FPU，MMUが搭載されている．空いているソケットには，Weitek社の浮動小数点アクセラレータを付けることができる．



Encore社のMultimaxは，並列コンピュータシステムの研究のプラットフォームとして，すでに米国や日本の研究機関で採用されている．特に，DARPAとのUltraプロジェクトにおける1000MIPSのプロトタイプマシンGigamax（従来Ultramaxと呼ばれていた）は，Multimaxを光リンクで8台結合したものであることは，よく知られている．すでにこのプロジェクトの目標値は3000MIPSに変更され（プロセッサには，Motolora社のRISCチップ88000を採用．約400MIPSのMultimaxが8台接続される），今年中にもDARPAに納入される予定である．また最近では，Oracle，Ingresなどのマルチサーバアーキテクチャを採用したデータベースを利用して，Multimaxを巨大なデータベースマシンとして採用しているところもあるようだ．

Multimaxがこのように高いMIPS値を実現できるのは，プロセッサの数に比例した性能の向上を実現できるためである．複数のプロセッサが独自のストリームを処理できるように，OSやI／Oが並列化されており，高速なバスアーキテクチャを採用して，システムのボトルネックにならないように設計されている．また，ソフトウェア開発においても，並列プログラミング言語がサポートされている．今回は並列処理という観点から，このMultimaxの特徴的な機能を探っていく．

ハードウェア構成

Multimaxには，NS32332を採用した300ファミリーとNS32532を採用した500ファミリーの2つの流れがあり，それぞれ**表1**に示す構成になっている．この表を見ても分かるように，システムのモジュール性が非常に高く，プロセッサ，I／Oチャネル，メモリなどがボードを追加するだけで簡単に拡張できる．

Multimaxの基本ハードウェアは，**図1**に示すバスに直結された5種類のカードから構成される．このように，複数のプロセッサが単一のメモリをバスを介して共有し，各プロセッサはローカルなメモリを持っていないプロセッサの結合方式を密結合と呼ぶ．この方式は，実装が技術的にも容易であり，安価であることから，Multimaxのほかにも商用ベースのマシンで採用されている．またMultimaxでは，プロセッサ間に親子関係や優先順位などがなく，OSやアプリケーションが空いているプロセッサで順次実行される完全対称型のシステムである．

さて，図のように共有バスを介してすべての資源をすべてのプロセッサやプログラムが共有することによって，どのようなメリットが生まれるのだろうか．まず考えられることは，各プロセッサが同一のメモリを共有しているために，プロセッサ間の通信にメッセージのやり取りがなく，高速に通信できるという点である．また，システムメモリはすべてのプロセッサに共有されており，プロセスをダイナミックにプロセッサに割り当てることができる．さらに，OSが負荷を平均的に分散させることができるので，重いプロセスが実行された場合でも，極端に性能劣化が起こらないなどが考えられる．しかし，複数のプロセッサを単一のバスに接続すれば，バスのトランザクションが増え，その処理能力に影響を与えることは明らかである．そのため，Multimaxのシステムバス(Nanobus)は，非常に高速なデータ転送を実現するとともに，いくつかの巧妙な機構を備えている．

表1　Multimaxのシステム構成

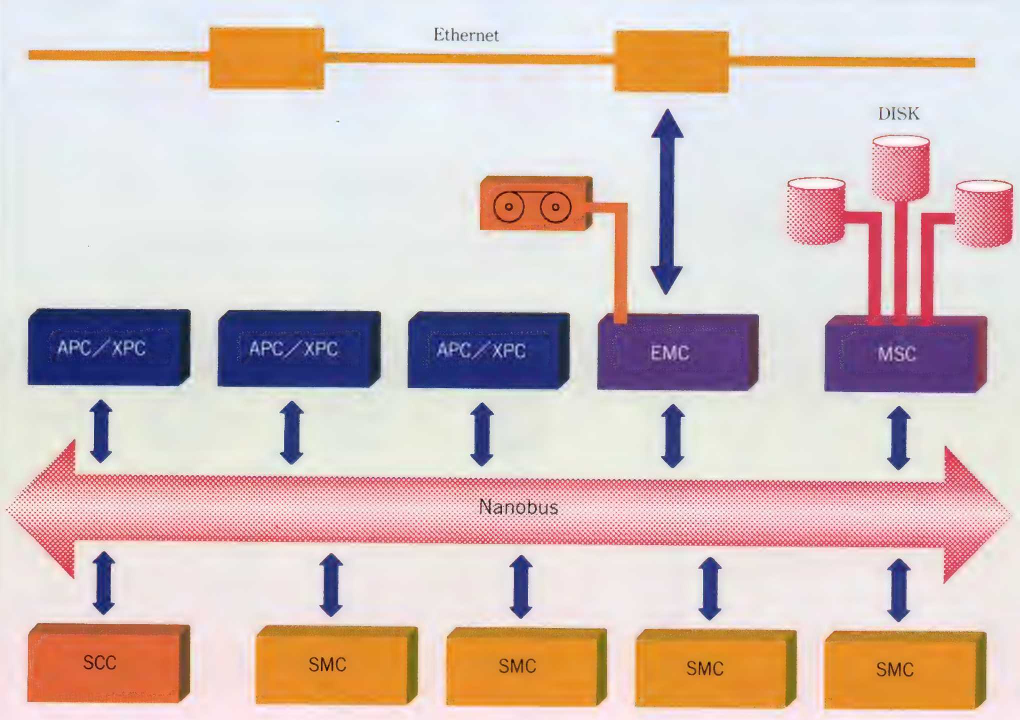
ファミリー	310	320	510	520
プロセッサ	32332	32332	32532	32532
プロセッサ速度	2MIPS	2MIPS	8.5MIPS	8.5MIPS
プロセッサ数	2～10	2～20	2～10	2～20
最大性能	20MIPS	40MIPS	85MIPS	170MIPS
キャッシュ	64Kbytes	64Kbytes	256Kbytes	256Kbytes
メモリ	16M～80Mbytes	16M～160Mbytes	32M～80Mbytes	32M～160Mbytes
ディスク容量	600M～9.6Gbytes	1G～67.2Gbytes	600M～9.6Gbytes	1G～67.2Gbytes
バスバンド幅	100Mbytes／秒			
ユーザー数	500＋	1000＋	500＋	1000＋

バスアーキテクチャ(Nanobus)

Multimaxのバスは，Nanobusと呼ばれる，バイポーラ同期バスを採用しており，100Mbytes／秒という非常に高いバンド幅でデータ転送が行なわれる．また，データバス(64bits＋8bitsパリティ)，アドレスバス(32bits＋4bitsパリティ)，ベクタバス(14bits)，制御ラインが分離されており，バスの占有率を分散させている（**図2**）．

Nanobusでは，最大80までの無関係のトランザクションを同時

図1　Multimaxのシステム・ダイヤグラム



・APC (Advanced dual Processor Card)
300ファミリーのプロセッサボード．NS32332を2個搭載し，各プロセッサごとに64Kbytesのキャッシュを備えている．

・XPC (eXtended performance dual Processor Card)
500ファミリーのプロセッサボード．NS32532を2個搭載し，各プロセッサごとに256Kbytesのキャッシュを備えている．

・SCC (System Control Card)
バスのアービトレーション，温度・電圧のモニタリング，ダイアグノスティック機能，コンソールはこのボードにつながっている．

・EMC (Ethernet/Mass Storage Card)
イーサネットとSCSI用に各1チャネル備えたI／Oチャネルボード．

・SMC (Shared Memory Card)
1枚当たり16Mbytesを実装したメモリカード．

・MSC (Mass Storage Card)
同期モードで動作するSCSIが3チャネル，1チャネル当たり4Mbytes/秒．

に扱うことができる．1つのトランザクションの完了は，他のトランザクションに依存しない．これは，Nanobusがペンディッド(pended)バスという方式を採用しているためである．通常のバスでは，プロセッサがメモリにリクエストを出した場合，リクエストが完了するまでバスを占有してしまうが，このバス方式では，リクエストは要求側のIDによってタグ付けされ（制御ラインを使う），すぐにバスは解放してしまう．そして受け側が後で応答を返すときには，その応答にも要求側のIDが付けられている．また，複数のバスのトランザクションをバッファリングして，異なるステージで処理を行なう，パイプライン処理も行なっている．

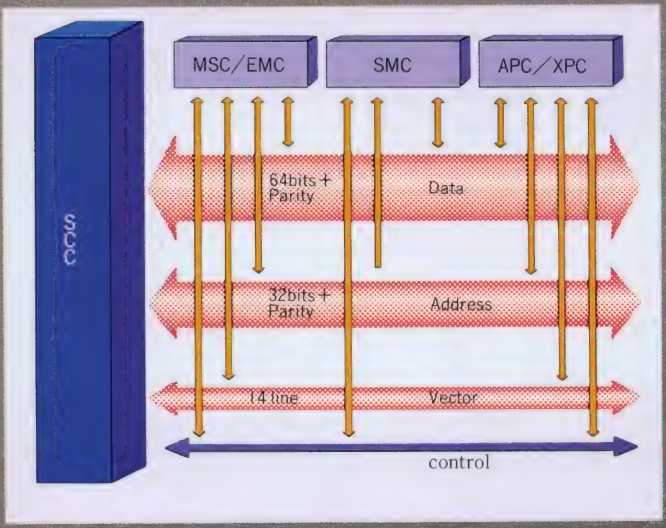
バスの高速性と比較すれば，メモリなどは比較的遅いデバイスである．そのため，メモリはインターリーブを行なって，アクセスの高速化を図っている．具体的には，SMCは2つの独立したバンクを持っており，個々のカードではバンク間で2ウェイのインターリーブを行ない，ボード間では4ウェイのインターリーブを行なっている．

キャッシュのメカニズム

複数のプロセッサが直接バスを介して主記憶をアクセスすると，すべてのプロセッサのアクセスがバスに集中し，システム全体のボトルネックになってしまう．また，共有バスを介するため，アクセスタイムが大きくなる．これらの問題を解決するために，プロセッサと主記憶の間に高速な記憶装置としてキャッシュが置かれる．

APC（Advanced dual Processor Card）では，1プロセッサ当たり64Kbytes，XPC(eXtended performance dual Processor Card）では1プロセッサ当たり256Kbytesのキャッシュ（どちらも25n秒のSRAM)が搭載されている．通常の単一プロセッサの場合は，キャッシュに書き込まれたデータを正確に主記憶に書き戻すだけですむが，マルチプロセッサの場合は，1つのデータのコピーが複数のプロセッサにコピーされるため，それらが最新のデータであることを保証するメカニズムが必要になる．バス結合型のマルチプロセッサでは，バスの状態をすべてのキャッシュが監視できるため，これをハードウェアで実現しているのが普通だ．Multimaxの場合は，キャッシュがバスタグ回路（BUS TAG LOGIC）という機構を用意しており，プロセッサ側からのアクセスをチェックするとともに，共有バス上のデータを監視している．このようなキャッシュをスヌープキャッシュと呼び，APCではライトスルー(write through)，XPCではライトディファー（write differ）というプロトコルを採用している．

図2　Nanobusバスの構造



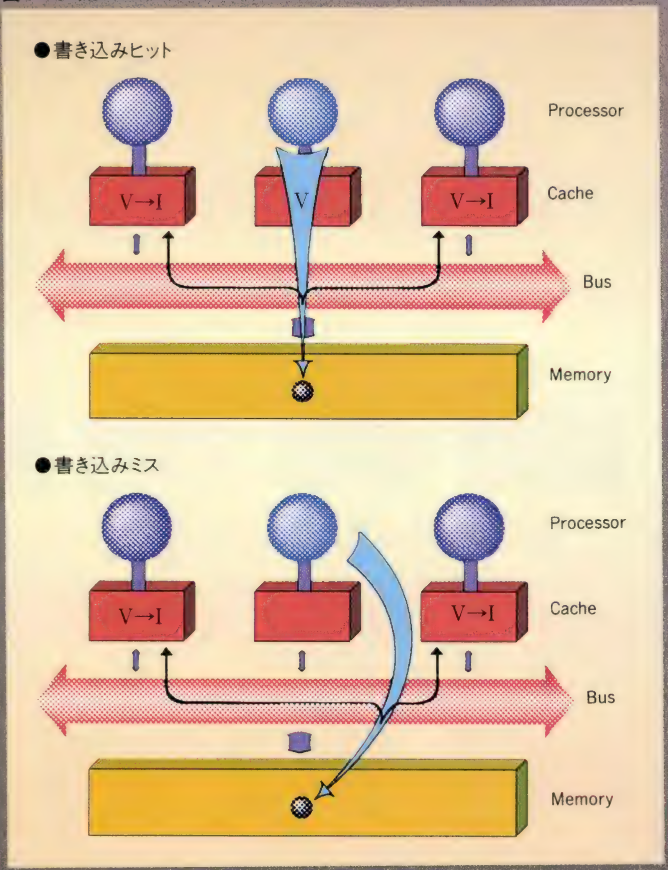
■ライトスルー

キャッシュと主記憶間でデータのやり取りを行なう場合，データの移動は，通常8bytesとか16bytesのブロック単位で行なわれる．ライトスルー方式では，各キャッシュのブロックに1bitのフラグを対応させ，INVALID（無効），VALID（有効）の2つの状態を持たせる．読み出し・書き込み要求によって，ミスした場合とヒットした場合で4つの状態が発生する．単一プロセッサと異なるのは，特に書き込み時の処理である．つまり，ヒットした場合には，共有バスを介して主記憶の更新を行なうとともに，他のプロセッサのキャッシュにそのブロックのコピーがある場合には，そのブロックをINVALIDにする．

■ライトディファー

ライトスルー方式では，すべての書き込みについて必ず共有バスのアクセスが伴い，バスの混雑により効率が低下する．そのためXPCでは，書き込み時にはバスをアクセスせずに，とりあえずコピー同士に矛盾があるままにしておき，後で必要になったとき

図3　ライトスルー



に主記憶に書き込む方式に変更された．

ライトディファーは，OWNED(所有)，SHARED(共有)，DIRTY(不一致)，INVALID(無効)の4つの状態によってキャッシュと主記憶の整合性を保つ，無効／主記憶書き戻し型のプロトコルである．プロセッサがキャッシュに存在しないデータを読む必要があるとき，キャッシュの回路は要求するアドレスに対してread publicコマンドを発行する．もし，他のキャッシュがそのブロックのコピーを持っていなければ，データを主記憶から取得して，キャッシュのデータをOWNEDとマークする．もし別のキャッシュがそのブロックのコピーを持っていれば，そのキャッシュはリクエストを認識し，そのブロックをSHAREDとマークする．要求側のキャッシュもまた，主記憶にアクセスして，そのブロックをやはりSHAREDとマークする．

プロセッサがデータを書き込む場合には，最初に主記憶のブロックに対する所有権を持たなければならない．もし，そのブロックがすでにほかのプロセッサに所有されたものであれば，キャッシュはread privateコマンドを発行する．その結果，ほかのすべてのキャッシュのこのブロックは，INVALIDとマークされる．そして，そのプロセッサは新しいデータをキャッシュに書き込み，書き込まれたキャッシュのブロックはDIRTYとマークされる．

このDIRTYとマークされたデータは，そのブロックがプロセッサによって再度書き込まれたり，別のプロセッサがread privateコマンドを発行したとき（この場合はINVALIDになる），あるいはそのブロックが，新しいキャッシュデータによって再配置されるまで（メモリに書き戻される）キャッシュ上に残されている．

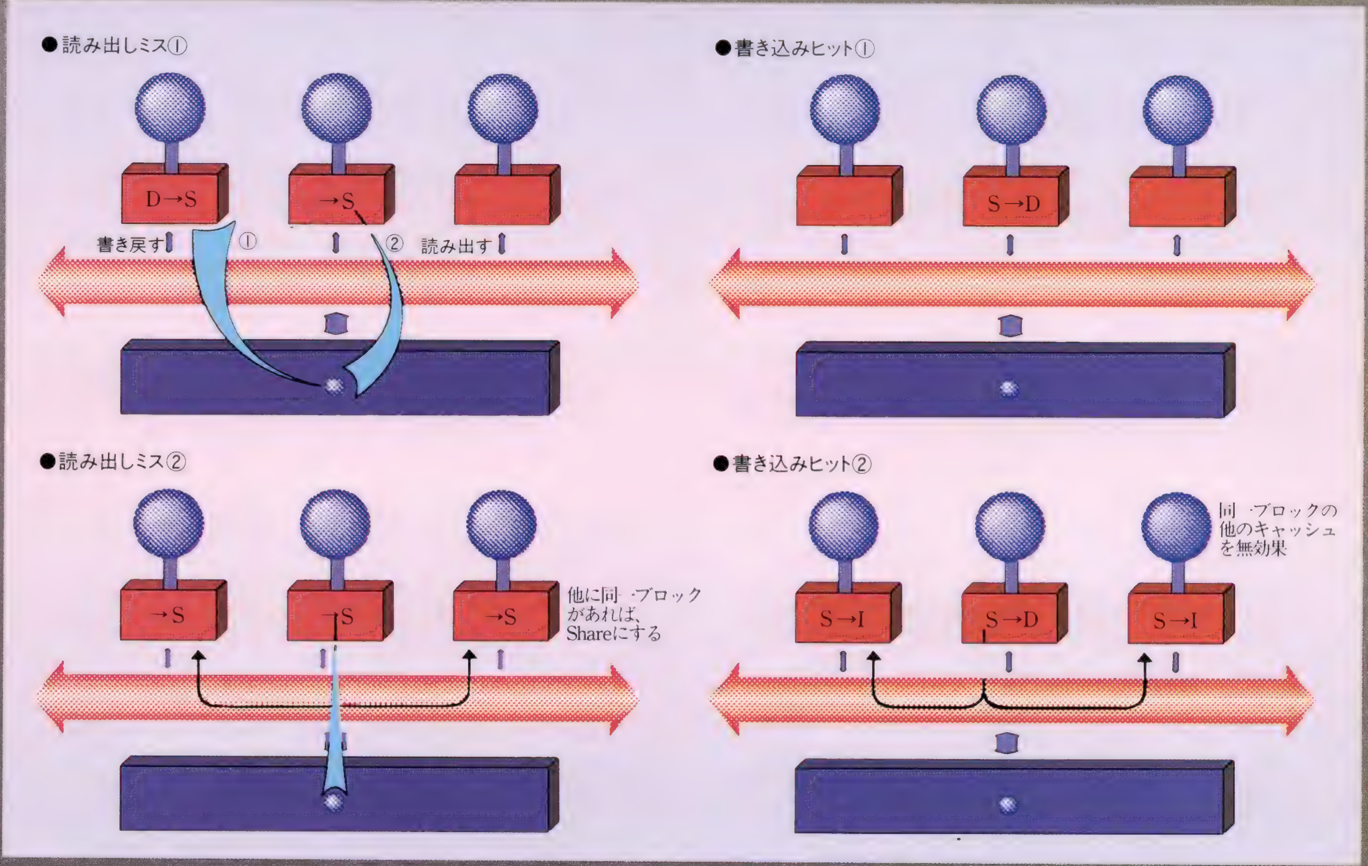
スヌープキャッシュのメカニズムとしては，ライトディファーは比較的単純なプロトコルであるが，MultimaxではUNIX系のOSをサポートしているため，各プロセッサでは独立性の高いプロセスが割り当てられ，キャッシュにはテキスト領域や変数領域が割り当てられる．これらの領域は，DIRTYかOWNEDのままだから，共有バスに対するアクセスは非常に少なくなり，このプロトコルでも十分に有効に機能することが分かる．

OSの機構

■スケジューリング

Multimaxでは，UMAX V (System V互換)，UMAX4.3 (4.3BSD互換)，Mach 0.5 (CMUのMach 2.5互換）の3つのOSがサポートされている．いずれのOSも，UNIXをベースにマルチプロセッサ対応に拡張したものだ．これまでの単一プロセッサ用に開発されたプログラムが実行される場合で，しかも“プロセス数≦プロセッサ数”であるときは，個々のプロセッサにプロセスが割り当てられる（実際には，時分割で実行されるので，最初から最後まで同一のプロセッサで実行されるわけではない）．そしてプロセスの数がプロセッサの数を超えると，各プロセッサでスケジューリングが行なわれ，時分割でプロセッサごとに処理は分割される．結果として，処理性能は（オーバーヘッドを除けば）プロセ

図4　ライトディファー



の数に比例して向上する.

■同期機構

このように，従来の構造のプログラムのままでも全体としてのスループットの向上を図ることはできるが，さらにプロセッサの空き時間を有効に利用するためには，プログラム自体のアルゴリズムを並列化しなければならない．たとえば，folkで子プロセスを生成して，データを分割して並列処理を行なう場合，最終的にデータを結合するためには，共有メモリを使用したプロセス間の同期機構が必要になる．UMAXでは次のような同期機構をライブラリで提供している.

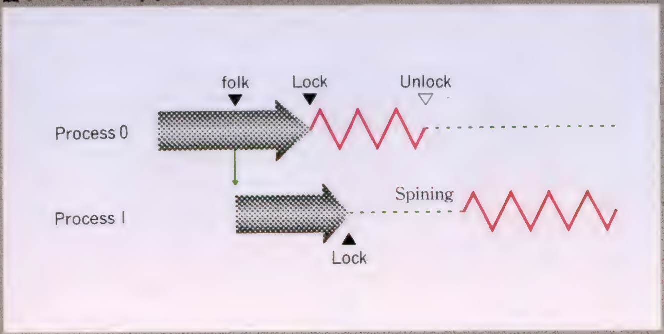
- ●スピンロック（Spinlock）
- ●セマフォ（Semaphore）
- ●バリア（Barrier）
- ●イベント（Event）
- ●モニタ（Monitor）
- ●リード／ライトロック（Read／Write lock）

この中では，セマフォとモニタはごく一般的な同期機構なので，ここでは説明を省略し，ほかのものについて簡単に説明しよう.

●スピンロック

複数プロセスによる同一共有メモリ領域へのアクセスを排他的に制御する．つまり，1つのプロセスが共有メモリをロックしている間は，他のプロセスがその共有メモリをアクセスしても，スピニング（空回り）させてアクセスできないようにする.

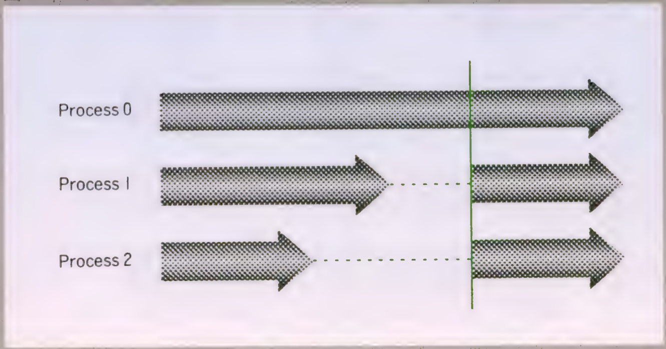
図5　スピンロック



●バリア

全プロセスの足並みをそろえるために，ある地点で全プロセスの到着を待ち，いっせいに処理を再開する.

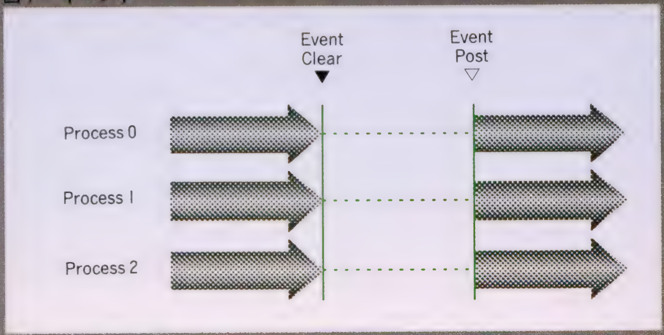
図6　バリア



●イベント

ある事象が起きるまで，複数のプロセスの実行を停止し，その事象が起きたときにプロセスを継続する.

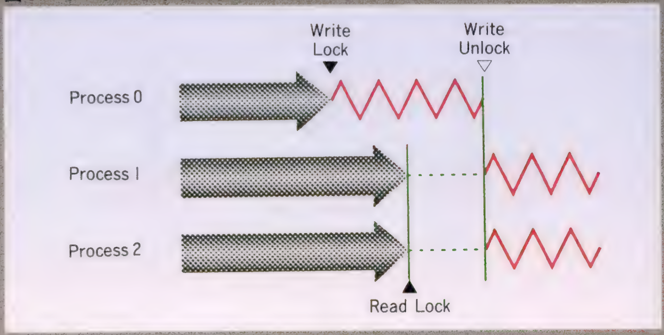
図7　イベント



●リード／ライトロック

セマフォの特殊なタイプであり，しばしば生産者－消費者問題として説明される機構である．この機構では，共有メモリに対する書き込み専用と読み出し専用の2種類のプロセスの存在が前提になっており，書き込みプロセスが共有メモリに書き込みを行なうときは，まずそのメモリにロックをかける．ロックされている間は，読み出し側のプロセスは共有メモリにアクセスすることはできない．書き込み側がロックを解除した状態で読み出し側は共有メモリをアクセスできるようになるが，その場合もやはり共有メモリをロックして，書き込み側がアクセスできないようにする．なお，読み出し側のプロセスの数は，複数あってもかまわない.

図8　リード／ライトロック



並列プログラミング環境

並列処理プログラミングの第1歩は，処理内容の並列性を発見することである．そのうえで並列に実行される仕事の大きさや数によって，アルゴリズムを工夫する必要がある．プログラミング言語はそれを表現する手段であり，並列化された仕事(プロセス)の相互関係をどのように定義するか，プロセッサとプロセスとの関係をどの程度明示的に指示できるかが，言語の能力を左右する．また，並列プログラムのデバッグの難しさを考えれば，プログラミング言語に留まらず，プロファイラ，デバッガなども備えた，総合的な開発環境が必要とされる.

Multimaxでは，並列処理プログラミング用の言語として，EPF (Encore Parallel Fortran)，Fortran 77，C，Adaがサポートさ

リスト1　EPFで記述した最大・最小・平均値を求めるサブルーチン

```
c
      subroutine find_min_max(a,n,shr_amin,shr_amax,shr_average)
      real a(n),amin,amax,sum,tmpsum
      real shr_amin,shr_amax,shr_average
      integer n,i

      shr_amax = a(1)
      shr_amin = a(1)
      sum = 0

      parallel
      private amin,amax,tmpsum,i

      amax = a(1)
      amin = a(1)
      tmpsum = 0

      doall(i = 1:n)
            if (a(i) .gt. amax) then
                  amax = a(i)
            endif
            if (a(i) .lt. amin) then
                  amin = a(i)
            endif
            tmpsum = tmpsum + a(i)
      end doall

      critical section
            if (amax .gt. shr_amax) then
                  shr_amax = amax
            endif
            if (amin .lt. shr_amin) then
                  shr_amin = amin
            endif
            sum = sum + tmpsum
      end critical section

      barrier begin
            shr_average = sum / n
      end barrier

      barrier

      end parallel

      return
      end
c
```

- shr_amax = a(1)～sum = 0：共有変数の初期化
- parallel：並列処理の開始．end parallel文までが並列に実行される．環境変数EPR-PROCSによって，作られるすべてのプロセス数が決まる
- private amin,amax,tmpsum,i：プライベート変数の宣言．EPFではprivate宣言しない限り共有変数となる．privateと宣言すると，並列領域でそれぞれのプロセスに変数のコピーが作られる
- amax = a(1)～tmpsum = 0：プライベート変数の初期化
- doall(i = 1:n)：doall～end doall文は，各々のプロセスにループ内の作業を配分するために用いる
- if (a(i) .gt. amax) then～tmpsum = tmpsum + a(i)：この部分の計算が添字ごとに分配されて並列に実行される
- critical section：critical section～end critical sectionの間の文は，排他制御され，一度に1つのプロセスでしか実行されず，ほかのプロセスはその間待たされる
- if (amax .gt. shr_amax) then～sum = sum + tmpsum：共有変数を更新する．このため排他制御が必要になる
- barrier begin：すべてのプロセスがbarrier begin文に到着した後，1つのプロセスのみが，barrier beginとend barrierの間の文の実行を許される
- barrier：すべてのプロセスが到着するのを待つ．end parallelには自動barrierの機能はない
- end parallel：並列領域の終了

れているが，新たな並列処理プログラミング環境として，Parasightという環境の開発も進められている．

現在利用可能な4つの処理系の中では，EPFによるプログラム開発が最も容易だといわれている．EPFは，Fortran 77を並列処理プログラミング用に拡張した言語であり，PARALLEL～END PARALLEL文によって，任意の数の並列処理部分を持つプログラムを記述できる（**表2**）．PARALLEL文が実行されると，いくつかの子プロセスが起動されるが，この数は環境変数EPRPRCSに設定された値によって決まる（EPRPROCS＝10ならば，9個のプロセスが作成される）．また，Fortran 77のプログラムからプリコンパイラを通して，DOループを並列化し，自動的にEPFプログラムを作成することもできる．EPFによる並列処理プログラミングの例として，最大・最小・平均値を求めるサブルーチンを示したので，各種の制御文の具体的な使い方を見ていただきたい．

ま　と　め

Multimaxの次のバージョンで88000シリーズが採用されれば，ハードウェアの性能は現在の2倍以上（およそ400MIPS）になることも夢ではない．プロセッサの性能向上と並列化を組み合わせれば，Multimax単体のシステムで1000MIPSを達成するのも，そう遠い先のことではないだろう．しかしソフトウェア環境の整備という点では，まだ課題が残されているようだ．密結合マルチプロセッサというアーキテクチャを活かしきる，OS，言語，デバッグ環境が，さらに充実される必要があるだろう．EPFのプログラム例を見ても分かるように，現状のコンパイラの仕様では，かなりプリミティブなレベルで並列性を明示的に記述しなければならない．プログラマの立場で考えれば，並列化が可能なところはすべてコンパイラで処理されるのが望ましいだろう．

最後に，並列処理機構および並列処理プログラミングに関する書籍を示しておいた．興味のある方は読んでいただきたい．

表2　並列プログラミング用の主な制御文

制御文	説明
PARALLEL～END PARALLEL	プロセスの生成と消滅．PARALLEL～END PARALLEL文は，プログラムの中で何度も記述できる．子プロセスの生成は1回目のPARALLEL文で行ない，END PARALLEL文に出会うとプロセスはアイドル状態にされる．2回目以降のPARALLEL文では，プロセスの生成は行なわずにアイドル状態のプロセスを実行可能状態に遷移させる
BARRIER	全プロセスの処理の進行を揃えるために，ある地点で全プロセスの到着を待ち，いっせいに処理を再開する
BARRIER BEGIN～END BARRIER	ある処理区間を1プロセスのみに実行させる
CRITICAL SECTION～END CRITICAL SECTION	ある処理区間（クリティカルセクション）へのプロセスの進入を一度に1プロセスに制限し，各プロセスはこの区間を交互に実行する
DOALL～END DOALL	DOループを各プロセスに分割し，並列に処理する
LOCK，EVENT	複数プロセスによる同一共有メモリへのアクセスを排他制御する
CONDLOCK	LOCK変数の値を返す
NTASKS	動作可能なプロセス数を返す
TASKID	自分のプロセスIDを返す

参　考　文　献

・『Multimax Technical summary』Rev. E January, 1989 Encore Computer Corporation

・『Introduction to Parallel Programming』Steven Brawer ACADEMIC PRESS, INC. 邦訳『並列プログラミングの基礎』大森健児訳　丸善株式会社刊

・『並列処理機構』高橋義造編　丸善株式会社刊

・『オペレーティングシステム』前川　守　岩波講座

PostScript Level 2の全貌 その2

松下電器産業株式会社
BE ワークステーションプロジェクト
宮部 義幸

前回は，今年6月の第1回Adobe Systems社（以下Adobe社）主催Developers' Conferenceでの話題を中心に最近のPostScript事情を紹介したが，今回は，これらをさらに深く掘り下げてみることにしよう．
初めてPostScriptに遭遇する読者のために，まず，PostScriptの概要について述べ，そのあとで，PostScript Level 2およびDisplay PostScriptを紹介する．

PostScript概論

昨今，我々のまわりのほとんどの文書が，電子的に作成されているといっても過言ではないであろう．

世の中には，一太郎，PageMaker，QuarkXpress，FrameMaker，Publiss，TEX，troff，はたまた，U1PRO，Rupo，OASYSをはじめとする多くの文書作成の道具が出回っている．さて，

「PostScriptは，このうちどれに一番近いものなんでしょう？TEXでしょうか？」

これは，少し的外れな質問である．PostScriptは，文書作成のアプリケーションソフトウェアと，プリンタ，ディスプレイなどのデバイスとの仲立ちをする，文書フォーマットなのである．

「つまり，SGMLやODAのような標準文書ファイルの一種なんですね？」

答えは，ある意味ではYesであるが，ある意味ではNoである．

PostScriptは，“ページ記述言語”であって，“共通文書ファイル”であるSGMLやODAのように，その主たる目的が複数の文書処理システム間で文書を交換しようとするものではなく，すでにでき上がったページのイメージを記述し，種々のアウトプットデバイスへの出力をサポートしようというものである．

写真1

DECによるDisplay PostScriptのデモ画面

図1に“ページ記述言語”と“共通文書ファイル”の違いを示してみた．

ここで，PostScriptの特徴を整理すると，以下のようにまとめられる．

- ●記述が解像度に依存しない．
- ●手続き，ループ，条件判断などの制御構造や，変数，配列などのデータ構造を持つ比較的高級な言語である．
- ●アウトラインフォントとそのインテリジェントなラスタライザを内蔵．
- ●アウトラインフォントのフォーマットが公開されており，多くの書体のフォントをダウンロードなどで，使用可能．
- ●直線，円弧，3次元曲線から構成される任意の図形の記述が可能．
- ●任意の図形を任意の色で塗りつぶすほか，任意の図形をクリッピングパスと呼ばれるステンシルとして使用可能．
- ●グラフィックスとテキストを同じ概念で操作できる．
- ●スキャナなどでデジタイズされた画像も，解像度にとらわれずに扱える．
- ●座標系の平行移動，拡大・縮小，回転，鏡像変換などがサポートされている．

PostScript導入のメリット

PostScriptは，通常，アウトプットデバイスに実装されたインタプリタで解釈される（**図1**）．そしてホストプログラムがPostScriptコードを出力し，インタプリタがイメージに組み立てる．

このように，あるアプリケーションは，PostScriptとホストプログラムの作業分担によって実現される．

たとえば，ホストプログラムがワードプロセッサのような場合，使用できる文字フォントの種類や，それらの拡大，変形，白抜きなどのいわゆる文字修飾に関わる作業に代表されるイメージ処理部分や，低解像度プリンタからプロ用写植機に至るまでのアウトプットデバイスサポートなどはPostScriptが担当してくれるため，ホストプログラムは，編集機能の向上に専念して開発できる．

現在，非PostScriptの文化のもとでは，多くの場合，アプリケーションごとにイメージ処理部分やアウトプットデバイスサポートなどのために，二重三重に開発コストや労力かかることを考えると，PostScriptは，開発側の負担軽減のみならず，ユーザーにとっても製品の低価格化，高機能化など多くのメリットを与えてくれる．

このような観点からPostScriptはページ記述言語であるとともに，グラフィックスアウトプットデバイスのためのオペレーティングシステムであるということもできよう．

Encapusulated PostScript File Format

前述のホストプログラムとアウトプットデバイス間の仲立ちとしてのPostScriptのほかに，アプリケーション間の共通データとしてのPostScriptがある．

もともとPostScriptはページ記述言語として開発されたものであるから，アプリケーション，すなわち，編集系間を自由に交換できるような性質のものではない．しかしながら，ある種の制約を与えることによって，どのような相手先へ持っていってもその編集系で，移動，拡大・縮小，回転などの加工を自由自在に行なって取り込めるPostScriptフォーマットを規定することができる．

これはEncapusulated PostScript File Format (EPSF) と呼ばれる規約で，同じくAdobe社の定めるPostScriptファイルの構造化規約Document Structureing Conventionから派生したもので，最新のものはAdobe社の電子メールによる情報サービスでも入手可能である（囲み参照）．**図3**にEPSFの非常に単純なサンプルを示す．また，EPSFの規約を以下に要約する．

●先頭に

%! PS-Adobe-2.0 EPSF-2.0

という行をつける．ここで，数字は規約のバージョンを示す．

●2行目の

%% BoundingBox:llx lly urx ury

で，そのファイルを描画したとき占める位置をデフォルトユーザー座標系で示す．

●EPSファイル外に操作が及ぶような手続き操作（スタックをすべてクリアするなど）を行なわない．

●EPSFが描画する図形のイメージを，必要であれば，付加できる．

●そのEPSファイルに関する種々の情報を %% に続くコメント行に記す．

図3　EPSFの例

```
%! PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 80 24
%%Pages 0
%%Creator: Glenn Reid
%%CreationDate: 19/9/1988
%%EndComments
%%BeginPreview: 80 24 1 24
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FF0000000000000000FF
% FF0000000000000000FF
% FF0000000000000000FF
% FF0000000000000000FF
% FF0000000000000000FF
% FF0000000000000000FF
% FF0000000000000000FF
% FF0000000000000000FF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
% FFFFFFFFFFFFFFFFFFFF
%%EndPreview
%%EndProlog
4 4 moveto 72 0 rlineto
0 16 rlineto
-72 0 rlineto closepath
8 setlinewidth stroke
%%Trailer
```

図1　共通文書ファイルとページ記述言語

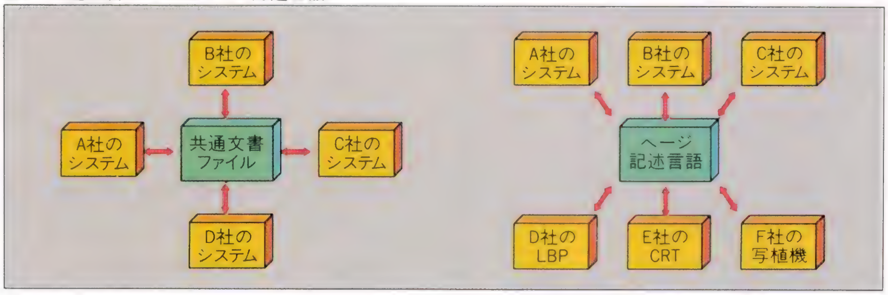
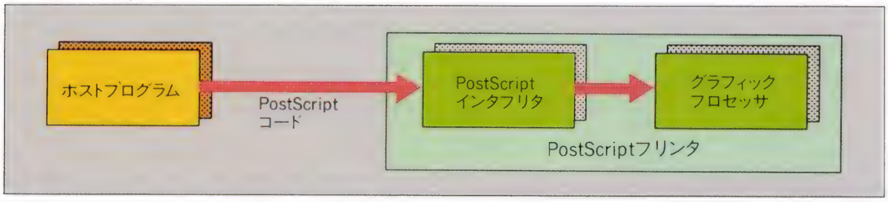


図2　PostScriptアプリケーションの構成



EPSFは異なるシステム間,特に解像度が違う入出力デバイスのシステム間のグラフィックスデータ交換（転入・転出）に絶大な力を発揮する．米国ではすでに，MS-DOSやMacintosh ,UNIX上の多くのソフトがEPSF対応である．

PostScrpitやEPSFは,マルチベンダのソフトウェア・ハードウェアによる文書処理システムを構築する上で極めて重要である．これらは,ISOのSGML関連のページ記述言語であるSPDLへの採用が検討されているほか，PanasonicのBEワークステーションで提唱されているBE-doff (BE Document File Format)などに採用されている．

PostScript Level 2

Adobe社は第2世代のPostScriptであるLevel 2の規格の発表を行なった.概要については前号で述べたが，今回はさらに詳しくふれてみることにする．

Level 2の特徴

Level 2の特徴は,

●Adobe社が別のシステムあるいはソフトウェアとして行なってきたPostScriptの改良や拡張をLevel 2に統合した.統合された技術は,

－Display PostScript(DPS)

スクリーンに高速に描画するために追加した新たなオペレータやアルゴリズム.

－Adobe Type Manager(ATM)

フォントの，特に小サイズでの，ラスタライズ技術.

－Composite Font Technology

マルチバイトコードの文字セットのためのComposite Fontの技術.

●カラーとイメージの処理が改善された．具体的には,

－イメージデータの圧縮形式がサポートされた.

イメージデータの表現は従来のPost Scriptでも可能であるが, Level 1の非圧縮ASCIIファイルではデータサイズが大きくなるため，ホスト上でのイメージデータの保存や，出力デバイスへの転送などで実用上問題がある場合も少なくなかった.

Level 2では,モノクロイメージに関してCCITTのGroupIIIとIVを，ハーフトーンイメージに関してランレングスエンコーディングを,カラー画像に関してJPEGを,7bitの通信チャンネルに対してASCII85を各使用できる.

今までは，カラーデータやイメージデータをPostScriptで扱うことをためらう場合もあったが，これにより，TIFFなどの，このようなデータに適したファイルフォーマットと併用することなく，すべてのデータをPostScriptで扱うことができるようになるであろう.

－デバイスインディペンデントなカラー表現をサポート.

Level 1でサポートしていたRGB, CMYKなどのカラー表現による画像は,デバイスの特性に依存して発色が異なることが多い. Level 2ではCIELABなどのデバイスに依存しないカラー表現をサポートしている.

－フォーム，パターンなどいくつかの機能強化が行なわれた.

よく使用するフォームの登録機能や,パターンをフォントのようにキャッシュする機能が追加されている.

Level 2のコンパチビリティ

PostScriptは,すでにデファクトスタンダードとして広く使われており，安易に言語仕様を変更することは得策でない.今回のLevel 2のスペックについてもこのような点が考慮され，最小限にとどめられてはいるものの，若干の言語仕様の拡張が行なわれている.

しかしながら, PostScriptは通常のプリンタ制御コードと違い，それ自身プログラム言語としての機能を持つので, Level 2で実施された言語仕様の拡張も大部分をLevel 1上のPostScriptプログラミングで吸収できる.

たとえば, Level 2で拡張されたrectfillというオペレータは,

llx lly width height rectfill

のように使える．これにより，長方形の塗り潰し速度をLevel 1と比較すると約3倍(Adobe社のデータ)となる．このオペレータを想定したLevel 2のアプリケーションの出力をLevel 1のプリンタでも印字する場合，次のようなエミュレーションプログラムで実現できる.

電子メールで入手可能なPostScript最新情報

Adobe社ではPostScriptに関するテクニカルドキュメントの配布サービスを行なっているが，これらの一部は電子メール（海外との電子メール交換ができる環境下にいる場合であるが）で入手することができる.

ps-file-server@adobe.COM

に,要求を出せばAdobe社のメールシステムが自動的に結果を返送してくれる.

残念ながら現在のところAdobe社はsunやdecwrlなどとのUUCP接続を持つのみなので，ftpでこのサービスを受けることはできない.

このシステムの使用方法であるが,

```
To: ps-file-server@adobe.COM
cc:
Subject:
---------
help
path 返信先のアドレス
```

のように, Subjectまたは本文にhelpと書いて先ほどのアドレスにメールを送ればヘルプメッセージが送られて来るので，これを参考にするとよい.

また，本文にpathという行を作って，そこにAdobe社から正しく届くようにアドレスを書いておくことが望ましい.

```
/rectfill where
 {pop /rf /rectfill load def}
 {/rf  {gsave newpath
 4 2 roll moveto 1
 index 0 rlineto 0 exch rlineto
 neg 0 rlineto fill grestore}
 bind def}  ifelse
```

このプログラムではrfというオペレータを定義して，このオペレータの内部でインタプリタがLevel 1であるか2であるかの判定を行ない，Level 1の場合は通常にカレントパスを構築して塗り潰しを行ない，Level 2ではrectfillで高速に塗り潰している．これのもとでは，

```
llx lly width height rf
```

とすることにより，Level 1でもLevel 2でも長方形が描画できる．このようなエミュレーションをアプリケーションのプリンタドライバ部分で実現しておけば，Level 1，Level 2を問わずPostScriptプリンタでの出力が可能となる．

Level 2の評価

さて，PostScript Level 2が実際に製品に搭載されて出荷されるのは1991年第1～2四半期といわれている．したがって，現時点ではLevel 2製品のパフォーマンスを実際には評価できない．

しかしながら，Level 2で実現されている機能の多くはDPSやATMから導入されたものである．これらは，現在我々が実際に手にすることのできるAdobe社の最新技術であり，これらを通してLevel 2の特徴の一部を垣間見ることができる．これらから分かることは，処理が非常に高速化されているということであろう．PostScriptは高機能であるが故に遅いというイメージがつきまとってきたが，一度DPSやATMを手にすると，そのイメージは完全に拭い去られるに違いない．これらは，解像度やハードウェアに違いがあるので一概にプリンタと速度比較することはできないが，対話的に使用してもほとんど問題が出ない速度が達成されている．

ATMは安価なソフトウェアでもあるので，一度手元のMacintoshで体験してみるのもよいだろう．

Level 2の中核 Display PostScript

高品位かつハイレベルなイメージングモデルを持つPostScriptをディスプレイ上で実現し，完全なWYSIWYGを実現するDPSは，カラーテクノロジーと並んで，PostScript Level 2を支える中核となる技術である．

以下，DPSについて，特にプログラミングの観点から迫ってみたい．

DPS プログラミング

DPSの使い方，すなわち，DPS上でアプリケーションソフトウェアを作成する方法について述べてみよう．そのまえに，DPSがどのようにワークステーションにインプリメントされているかを知る必要がある．

図4は，X Window Systemの拡張プロトコルを使用して実装されているDPS(以下DPS／Xと表記)のブロック図である．DPS／Xの場合，画面上のボタン，メニューなどのユーザーインターフェイス部分をX Windowに，テキスト，グラフィックス，イメージの操作にはDPSを，というように1つのアプリケーションに対して，X WindowとDPSを同時に使用することも可能である．

この点で，DPS／XはNeXTのDPSと若干異なってはいるが，イメージングモデルはまったく同一であり，両者間のソフトウェアの移植性は極めて高い．

さて，DPSを使うには定められたプロトコルに従ってDPSサーバにPostScriptコマンドを送り込めばよい．DPSはプリンタのPostScriptと違い，ユーザーとの直接対話の部分を担当するため，より高速な動作が要求される．

そのためサーバへ送るPostScriptコマンドもコンパクトさが要求される．プリンタPostScriptの場合，いかなる通信路でも確実に通り，しかも人間が見て分かりやすいASCIIコードの文字列でコマンドを表現するが，DPSではバイナリエンコードされたコマンドを送ることになっており，通信路におけるオーバーヘッドの軽減が図られている．

アプリケーションプログラマがこのバイナリエンコードされたコマンドをDPSサーバに送る便宜のために，クライアントライブラリと呼ばれるC言語の関数群が用意されている．DPSコマンドに1対1に対応したクライアントライブラリの関数をコールすることにより，該当コマンドがDPSサーバに送られる．

クライアントライブラリの使い方

クライアントライブラリの使用方法について，簡単なPostScriptプログラムを例にとって見てみることにしよう．

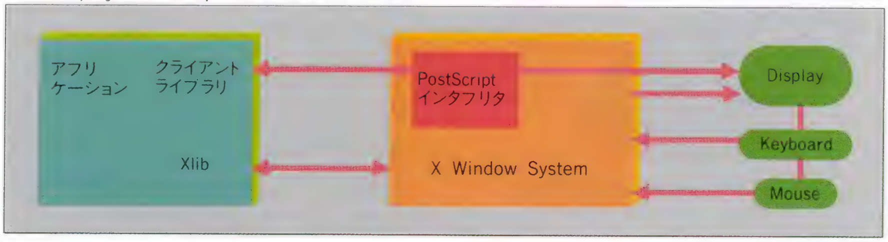
さて，**図5**の円グラフの部分の表示を行なうためにPostScriptでは，

```
205 200 moveto
205 200 100 315 45 arc
closepath
195 200 moveto
195 200 100 315 45 arcn
closepath
fill
```

と書けばよいことはよく知られている．

これを DPSのクライアントライブラリを使用して書くと，

図4 Display PostScriptのインプリメンテーション



```
void PieChart(void)
 {
 PSmoveto(205.0, 200.0);
 PSarc(205.0, 200.0, 100.0,
   315.0, 45.0);
 PSclosepath ( ) ;
 PSmoveto(195.0, 200.0);
 PSarcn(195.0, 200.0, 100.0,
   315.0, 45.0);
 PSclosepath ( ) ;
 PSfill ( ) ;
 }
```

のようになる．これから分かるようにクライアントライブラリの関数はPostScriptのコマンドにほぼ1対1で対応している．このようにクライアントライブラリはほとんどPostScriptと同じ感覚で使用することができる．もちろん，座標系はPostScriptのデフォルトユーザー座標系，すなわち，1ポイント(＝1/72インチ)を単位とした左下が原点の座標系である．したがって，X Windowなどのように画面のサイズや画素数を意識したプログラミングを行なう必要はない．

また，対応するPostScriptコマンドをプリンタに送れば，ディスプレイと同じイメージのハードコピーを得られる．

次に，テキスト部分の表示を行なってみよう．PostScriptでは，

```
/Times-Roman findfont
24 scalefont setfont
110 80 moveto
(Display PostScript) show
```

すればよかった．これをクライアントライブラリで書くと次のようになる．

```
void ShowDPS (void)
 {
PSselectfont("Times-Roman",24.0);
PSmoveto(110.0, 80.0);
PSshow("Display PostScript");
 }
```

ここで，PSselectfont ( ) はfindfont, scalefont, setfontの3つの機能を1つにまとめたものである．

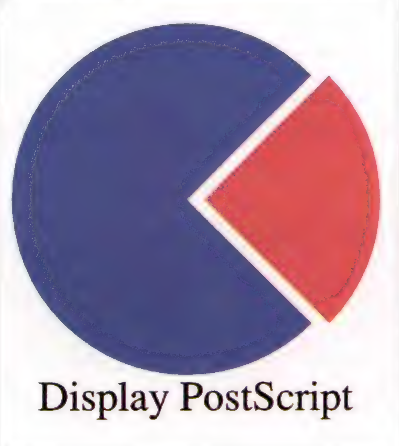
このように，DPSではサーバとの通信量を減らすことなどを目的に，一部新しいコマンドになっているものもある．PostScriptにおけるfindfont，scalefont，setfont という3つのオペレータを使用したフォント選択は，PostScriptの内部のメカニズムを知らないユーザーにとってみれば非常に不可解な感じのするものであったが，DPSにおけるPSselectfontはこの点，すなわちアプリケーションプログラマとのインターフェイスにおいても改善されているといえる．

PSWRAPを使う

C言語からのDPSの使い方にpswrapを使用する方法がある．これは，PostScript言語をC言語の中にラップする(包み込んでしまう) というものである．1つ具体例を示そう．先ほどの “Display PostScript ”というテキストを表示するプログラムを，pswrapを使う方法で書くと次のようになる．

```
defineps ShowDPSw( )
/Times-Roman findfont
24 scalefont setfont 110 80 moveto
(Display PostScript) show
endps
```

図5 サンプル出力



このプログラムをたとえばShowDPSw.pswというファイルにセーブしてpswrapで処理すれば，前述のShowDPS( )とまったく同じ結果を得るC言語の関数ShowDPSw( )ができ上がる．

pswrapは，PostScriptで書かれたコードをC言語の関数に変換するフィルタである．拡張子が.pswであるようなファイルをpswrapで処理するようにMakefileにでも定義しておけば，C言語とPostScriptを自由に使い分けて効率よくプログラムの開発ができるようになる．

pswrapを使用する場合にも，

```
defineps ShowStr(float X,Y; char*Str)
/Times-Roman findfont
24 scalefont setfont
XY moveto
Str show
endps
```

のようにC言語からの引数をPostScriptに与えることも可能である．

●　●　●　●　●　●

2回にわたってPostScript Level 2とそれを取り巻くAdobe社の新技術についてふれてきた．今後ワークステーションの低価格化などから，ビットマップディスプレイの一層の普及が期待され，アプリケーションプログラムもそのユーザーインターフェイスや印刷出力のビジュアル化がますます進むと予想される．そのとき，PostScript Level 2やDPSは，オペレーティングシステムやウィンドウシステムに次ぐ第3のコアソフトウェアとなるに違いない．

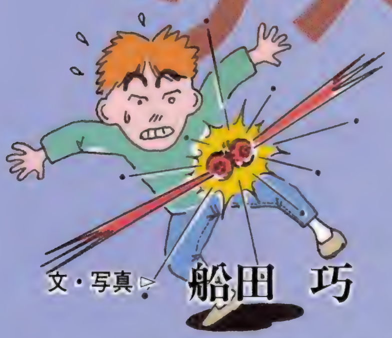
出力の多彩化と，これをフルに活かした高機能のエディタにより，デザインセンスにたけた人々の活躍の場がコンピュータの世界に広がることは間違いない．

しかしながら，たとえばWingzのような，一般の人々にも簡単に見映えの良い出力が得られる優れたソフトウェアが多数出現してはじめて，PostScriptが身近なものになるのではないか．

最先端

〈連載4〉

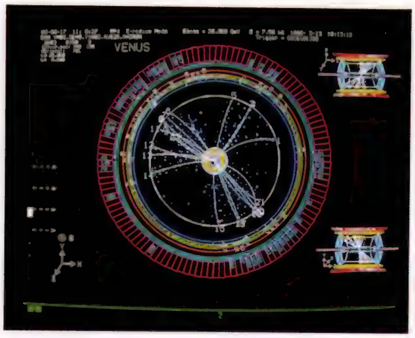
システム探検隊



高エネルギー物理学研究所の巻

文・写真▷船田 巧

うわさには聞いていたけど，ブワッと広い敷地に巨大な加速器，そこいら中にケーブルとゴツイ機械とコンピュータがゴロゴロ．とってもパワフルな研究施設に興奮してきました．



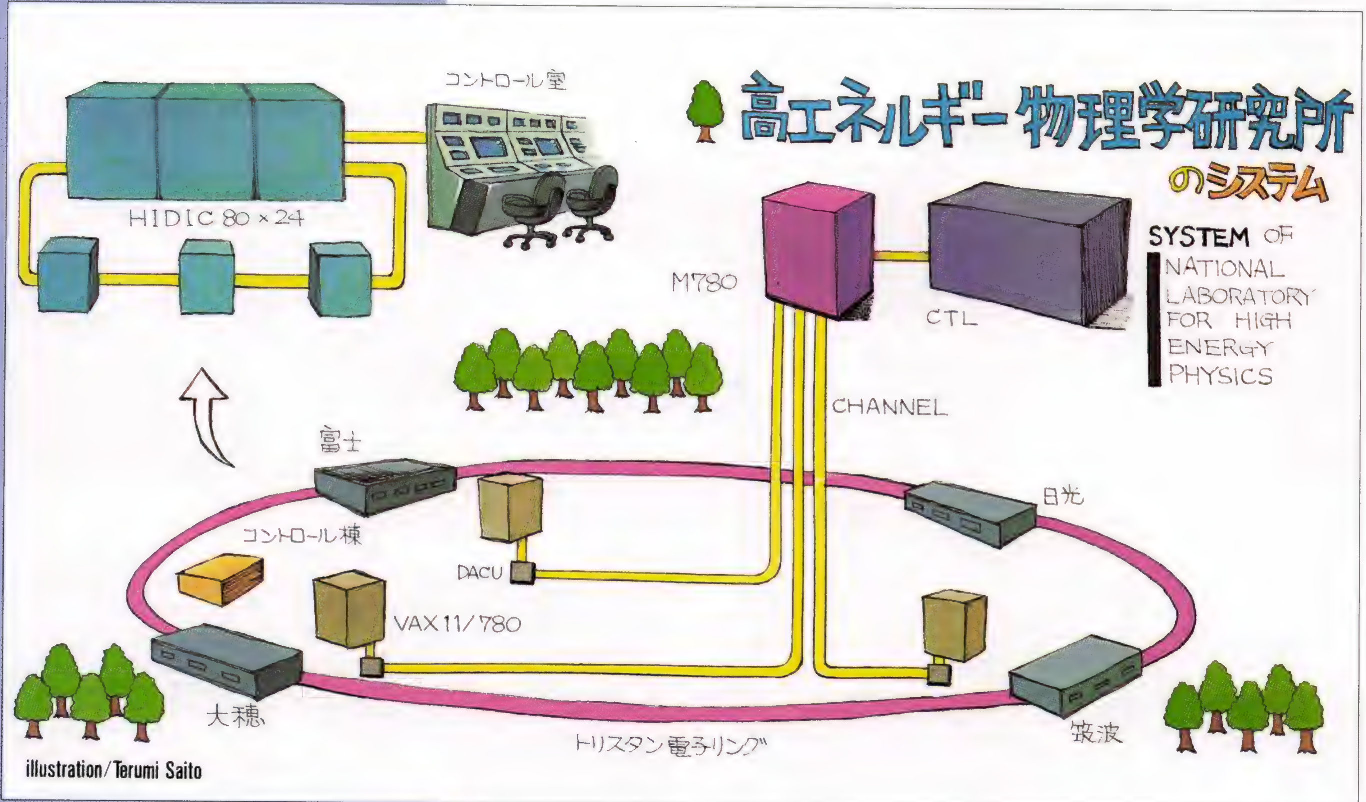
南青山から筑波研究学園都市まで自動車で1時間ちょっと．編集者兼運転手が鈴木保奈美似の美女だったらどんなにか楽しいドライブだろうなどと妄想しながら訪れたのが，文部省高エネルギー物理学研究所（KEK）である．

最近の宇宙論流行りもあって統一理論がどうのクォークがどうしたといった話を皆さんよく耳にしていると思うが，そういう素粒子の世界を実験をもとに探索していくのが，高エネルギー物理学だ．未発見の“トップ”クォーク，ヒッグス粒子，次世代のレプトンとクォークの存在を実験的に確証するのが目下の研究課題とされている．

そのための実験施設の1つとして昭和61年に完成したのがトリスタン加速器．トリスタンは全長3000mのリング中を，時計回りに電子，反時計回りに陽電子を走らせ，4カ所の実験室にある測定器内で衝突させて，その素粒子反応を観測するのが目的．簡単に電子と陽電子を衝突なんていってるが，車がないと回れないくらい長大なリングの中を，それぞれ30GeV (Giga electron Volt) まで加速しておいて，エンピツの先っちょと先っちょをくっつけるくらいの精度でぶつけるというのは土木工事の技術からしてたいへんなものである．当然，システムの制御にはコンピュータが不可欠．

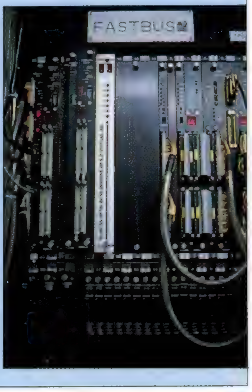
電子と陽電子が衝突したときに発生する粒子1つ1つを記録するのに用いられる測定器は現在3基．大穂実験室のAMYは700t，富士実験室のVENUSと筑波実験室のTOPAZは2500tもある巨大なもので，測定器内でのデータ変換，得られたデータの解析処理ともに，計算量もまた巨大である．

今回はトリスタン関係の計算機環境だ



illustration/Terumi Saito

トリスタン・コントロールシステムの制御室．コンソールの前の人物は加速器研究部の黒川真一教授．写真に写っている範囲が，1人分のコンソール．操作は手元のタッチスクリーンで行なう．奥のモニタにグローバルな情報が表示される．



CAMACのほかにFAST BUSと呼ばれる標準規格もある．これはTOPAZ測定器のAD変換部で使われていた回路．

けに絞って取材したのだが，それでも，制御系，データの収集解析系の独立した2つのシステムが存在している．どちらも大規模かつ面白い．まずは，制御系のトリスタン・コントロールシステムから見ていこう．

加速器は日立のHIDIC80というミニコン24台のネットワークで制御されている．HIDIC80は256Kwordのメモリ，17Mbytesのディスク装置を持ち，処理能力は約1MIPS．各機は10Mbpsの光ファイバでリング型に接続されており，さらにそれとは別にタイミング同期用のケーブルが張られている．24台のうち9台は制御室裏のマシンルームに設置され，コンソール，アラーム，ライブラリ（ゲートウェイ）などの管理，ソフトウェアの開発に使われる．残り15台は加速器各部に設置され，CAMAC という高エネルギー物理学分野の標準インターフェイスを介して3万にのぼる機器を制御している．

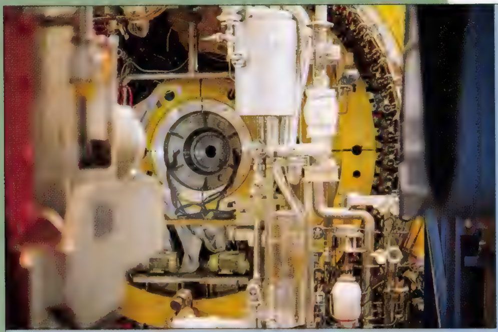
担当者が面白さを強調していたのが，トリスタン制御専用言語KEK NODALの存在．KEK NODALはインタプリタで，遠隔操作を行なうためのマルチコンピュータプログラミングが簡単にできること，ダイナミックなデータモジュールと関数ライブラリの管理によってハードウェアの変更にも容易に対応できることが主な特徴だ．

ここでいうマルチコンピュータプログラムっていうのは，プログラムそのものを送信しあい，それを各コンピュータのインタプリタが解釈，実行するという直感的に理解しやすいもので，「これならキミでも作れるでしょ」という意味かどうかは分からないが，渡されたマニュアルを読むと，なるほどプログラミング自体は分かりやすそうだと感じた次第．インタプリタは信頼性の面でも有利とのことであった．

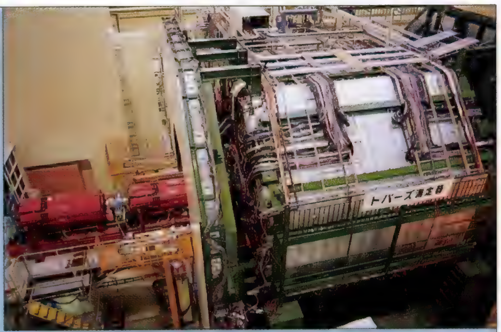
さて，さっそうとした制御室から地下へ潜ると，そこはゴツイ機械とパイプとケーブルの巣窟．電子と陽電子がぶつかるチェンバーからは3万チャネルに及ぶ信号を伝えるためのケーブルがぞろっと這い出し，プリアンプを経てADコンバータへと向かっている．ここがデータ収集，解析システムの入り口．

測定器からは最大500Kbytes／secのデータが発生する．衝突の発生をイベントというが，イベントは毎秒およそ5回でVENUS, TOPAZ測定器では1イベントあたり平均10Kbytes, AMYではその3倍程度のデータが発生する．

測定器からのデータは実験室のVAX11／780で圧縮され，IBM DACU (Device Attachment Control Unit) を仲立ちに



VENUS測定器のドリフトチェンバー入り口．写真手前から陽電子，反対側から電子が入り，中で衝突し，その反応を観測する．巨大だが繊細なセンサの塊．



上から見たTOPAZ測定器．2500tの重さで沈まないよう，レールに乗せて荷重を分散し，実験室の床は地下40mの安定した地盤まで届くパイルの上に乗っている．

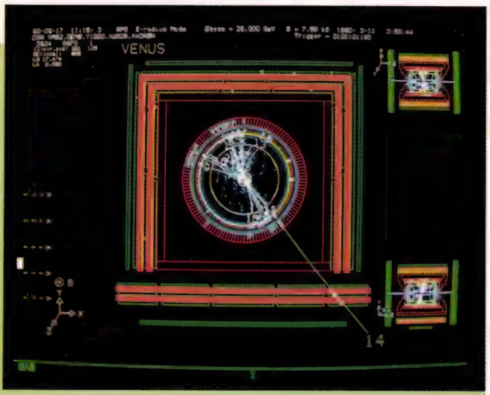
VENUS のハット．シガニー・ウィーバーとエイリアンが出てきそうだ．ここで，測定器からの信号が増幅，AD変換される．この上には冷却装置などがある．

VAX 11／780 の前に立つVENUSグループの坂本宏氏．実験室での測定データの前処理を行なうために，トランスピュータ（INMOS T800）を36個並列にした処理装置を開発中．

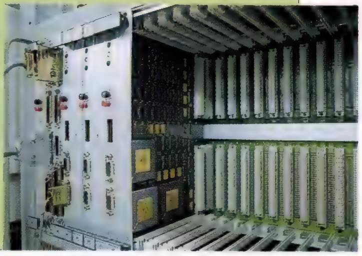
1キャビネット(横に長い)あたり1Tbytesを扱えるカートリッジテープライブラリ装置．手前がこれから装塡されるテープの山，後ろが記録済みのテープの山．

データ処理センターの渡瀬芳行教授．トリスタン用の富士通M780以外にも，多目的に使われている日立のスーパーコンピュータやメインフレームがある．

観測データの1つをグラフィックス表示したもの．中心の円がドラフトチェンバー，外側の枠がカロリーメーター．μ粒子のトラック(軌跡)がびゅんと出ているのがよく分かる．



TOPAZグループではモトローラの88000シリーズを使った並列プロセッサを作っていた．会った人みんながコンピュータ屋さんだったような気がする(物理出身の人がほとんどらしい)．



富士通M780へと送られる．M780で3つの測定器から送られたデータを各実験グループごとに解析する．

入ってくるデータは即時的に解析され，粒子1つ1つの種類，運動量，放射角度，エネルギーが求められる．そのためにFORTRANで40000行のプログラムが衝突の数かける実験グループ数だけ実行されるわけだが，今のCPUパワーではこれ以上データを増やしたり，精度を上げたりすることが難しい．スーパーコンピュータにしちゃえと，単純に思ったが，解析プログラムとの相性や運用形態の点であまりいい解決策ではないという話．現在進められているのは，CPUパワーの大きいワークステーションでイベントごとに分散処理する方法(GWSのIrisが計算だけに使われているらしい)と，VAX11/780へ入る前の段階でパラレルプロセッサによる前処理を行ない，ホストの負担を軽減するという方法が考えられている．

効率を重視して，トリスタンは数カ月単位に24時間連続運転され，1年間に3Tbytesものデータを発生する．この膨大なデータの保存はカートリッジテープライブラリ装置で行なわれる．常用20000本，着脱1日500回という人手では気の遠くなるような作業をロボットが黙々とこなしていく．

こうして蓄えられるデータが，素粒子の世界を知るうえでの基礎データとなるわけだが，最終的にそのデータは統計処理されて理論計算により予想されるデータと比較される．

たとえば，“トップ”クォークが存在する場合の衝突はこんなふうになるというのを，モンテカルロ法で計算したものがあって，それを実験から得たデータと比較し，“トップ”の存在を判断する．我が探検隊が取材に行ったときはちょうど，それまでの実験によってトリスタンのエネルギーレベルでは“トップ”の存在を確認できるデータは得られなかった，という結論が出た時期だった．

“トップ”探しは，派手なところを狙った実験だったが，今後は精密測定を行いますと笑いながら語っていたが，本当のところはかなりがっかりしていたのかもしれない．KEKで新しい素粒子が発見されますようにと，助手席で祈りつつ，筑波を後にした探検隊だったのでした．

オンライングループの藤井啓文助教授．手前のラックに刺さったボードを介して，VAXからのデータがM780に送られる．金属箔で包まれたフラットケーブルを見て富士通の技術者には内緒と言った．

サービスで表示してくれた測定器のCGがかっこいい．棚の上に立てられたキーボードがちょっと変だが，それを打っている姿もけっこう変であった．

今回はいつにも増して写真が多いこのページである．最後のカットは題して『トリスタンの休日』．主リングの西端より北へ向かって．

Astro Guide '90

火星探査の新時代

『21世紀の最初の10年のうちに，人類は再び月に帰る．そして今度はとどまる．ここからさらに，別の惑星，まず火星への有人探査を開始する』——昨年の7月20日，月着陸20周年を迎えたアメリカで，ブッシュ大統領は火星有人探査を宣言した．'61年のケネディ大統領ほどの影響力があるかどうか疑問は残るが，今回はヨーロッパ，ソ連，そして日本と世界を巻き込んだプロジェクトになるようだ．

その火星が11月20日に地球に接近する．これは'88年の大接近以来2年2カ月ぶりのことだ．ということで今月は火星に注目してみよう．

現在，我々が持っている火星の知識は'76年のバイキング1号，2号によるものが大きい．夏の暑い最中に火星着陸船から送られてきたカラー映像は今でも頭の中に鮮明に焼き付いている．

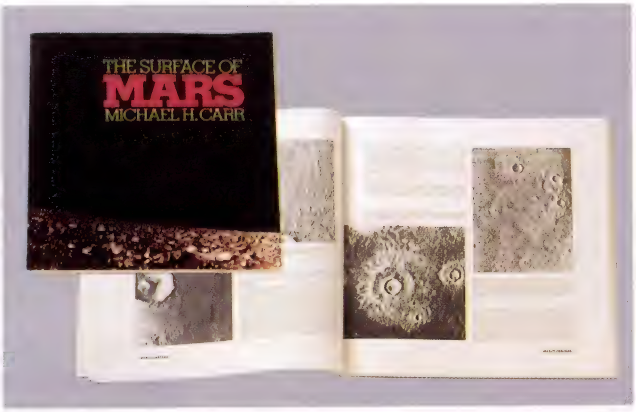
この2つの探査機で得た情報量は膨大なもので，いまだに世界の惑星学者たちが分析を続けている．火星の人面像など門外漢を楽しませてくれるものも多いが，公式にまとめられたものとしては，バイキング探査機イメージングチームのリーダーを務めたMichael Carr博士の『THE SURFACE OF MARS』が最も詳しい．火山，クレータ，渓谷など地形別に分けられ，その成因などが100枚を超える写真とともに解説されている．火星は惑星形成の原始の形をとどめたままで，そのダイナミックな風景には驚かされる．特に興味をそそられるのが水による浸食の跡で，かつて火星に海や川，湖があったことを思わせる．Carr博士によると，かつて火星にあった大量の水は，表面下1kmから2kmにわたる地下に永久凍土として残されているらしい．火星の風景を衛星から撮影した地球の映像と比べてあれこれ考えてみるのもまた楽しいものだ．

次の火星探査計画は，ソ連が'94年に打ち上げを予定している『火星94号計画』だ．これには，日本，フランス，イギリスなど合計12カ国が参加し，これまでにない大規模な国際研究プロジェクトになる．ローバーや気球などさまざまな観測機材を送り込み，火星の総合調査を行なう．このプロジェクトの推進に大きな役割を果たしたのが，Carl Sagan博士が会長の『Planetary Society』だ．同会は，世界最大の民間宇宙開発推進団体で今回のプロジェクトに関して世界から募金を集めている．あなたも募金を通じて火星探査計画に参加してはいかが．

(haru-k)

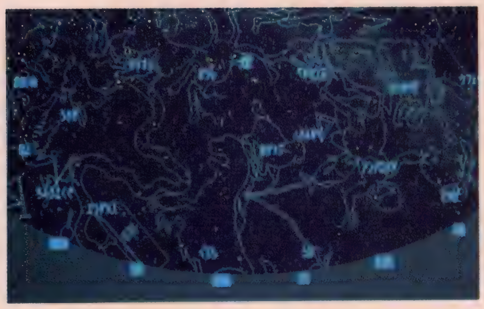
『The Surface of Mars』
Michael Carr
New Haven and London : Yale University Press,

Planetary Societyについての問い合わせ先：　矢沢事務所
☎ 03(423)2255



「THE SURFACE OF MARS」．写真はすべてモノクロながら，100枚以上の豊富な火星表面の写真に合わせて，詳しい解説がなされている．当然ながら本文はすべて英語であり，厚手の本にぎっしりと写真と文が詰まっている．これ1冊を読破すれば，火星博士になれるだろう．

今月の星空



11月20日の空

11月20日に接近する火星は，おうし座のプレアデス星団の付近にいる．赤い星アルデバランと好対照をなし，明るさも−2.0等なので都会でも簡単に見つけることができる．前回の'88年には5881万kmまで接近し大接近となったが，今回は7733万kmと中接近となる．これだけ接近すれば火星表面を十分楽しむことができる．もちろん地上からクレータなどは見えないが，極冠の変化や大シルチスなどアマチュア天文家の興味の対象には事欠かない．口径の大きな望遠鏡で見れば，時折火星の朝の部分で青白く朝靄がかかっている様子を見れるだろう．

第4回
ニュースシステム

プロフェッショナルワークベンチとしての
UNIX入門

村井 純

illustration/Yoshinori Otake

今回のテーマは，電子ニュースシステムです．

そもそも電子ニュースシステムは，ネットワーク上でコンピュータを利用している人がみんなで見られる掲示板のような仕組みを作ろうということから始まりました．電子メールを使ってもメッセージのやり取りはできるわけですが，個人と個人の間ではなく，1人が書いたことをみんなが見れるようにして，あたかも町内会の掲示板のように使えるようにしたものです．

関心あるトピックスの情報を得たり，自分が持っている情報を掲示板に貼ったりといったことをコンピュータネットワークシステムの中で実現するのがこの電子ニュースシステムなのです．

ニュースシステムの成り立ち

■ニュースシステムとネットワーク

パソコン通信に詳しい人であれば，この掲示板の感覚はよく分かると思います．しかし，皆さんが使う電子ニュースシステムというのは，パソコン通信よりももっと国際的なものです．SFCにあるワークステーションは，前回の電子メールのところでも説明したとおり，世界的なネットワークにつながっていますから，電子ニュースシステムもネットワークを利用して，さらにほかのところにある掲示板の情報をやり取りする仕組みを持っているのです．

慶應大学湘南藤沢キャンパスには‘sfc’という掲示板があります．さらに慶應大学全体の掲示板があります．これが‘keio’という掲示板ですが，これは慶應大学全体で見ることができます．

さらに日本全体の掲示板‘fj’があります．具体的には，前回説明したJUNETにつながっているコンピュータから見える掲示板です．

それから，世界全体の掲示板がありまして，これには‘comp’, ‘misc’, ‘alt’などといった名前が付けられています．

■ニュースグループの構造

それぞれの掲示板は，“ニュースグループ”と呼ばれるトピックスごとの掲示に分かれています．このトピックスは，そ

れぞれ階層的にグループ分けしてあります（図1）.

fjの場合，たとえばこの中に‘rec’というレクリエーションに関するニュースグループがあります．初めはレクリエーションのこと，研究や仕事以外の楽しみはすべてここに書き込むことにしていたのですが，そのうちスポーツのことを書き込む人もいればテレビのことを書き込む人もいて，混乱してきました．レクリエーションとひと言でいうよりももっと分化したほうがよいだろうということになり，recの中でさらにトピックスを分けていきます．テレビが好きな人には‘tv’というニュースグループを作り，スポーツが好きな人は‘sports’というニュースグループを作ります．さらにスポーツの中でも，スキーやフットボールの好きな人がいますから，‘ski’とか‘football’といったグループに分かれていったりします．

ニュースグループの名前は，これらをピリオドで結んで，‘fj.rec.sports.ski’などというふうに書きます．

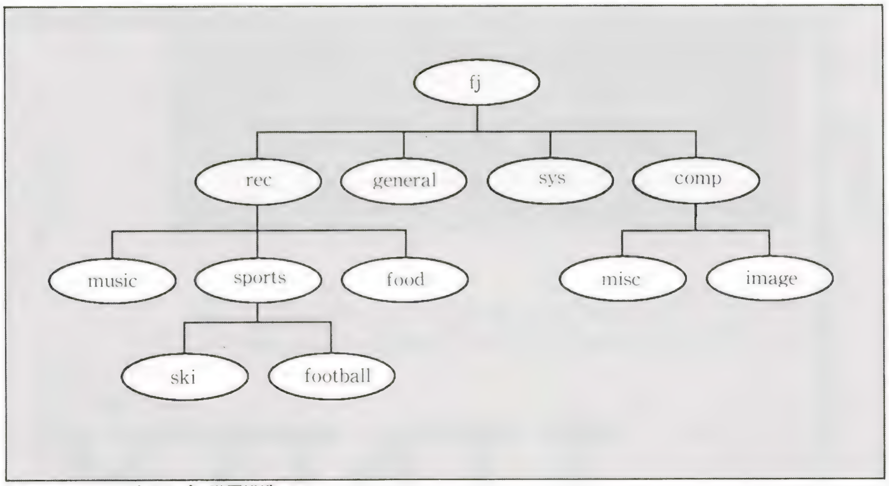


図1　ニュースグループの階層構造

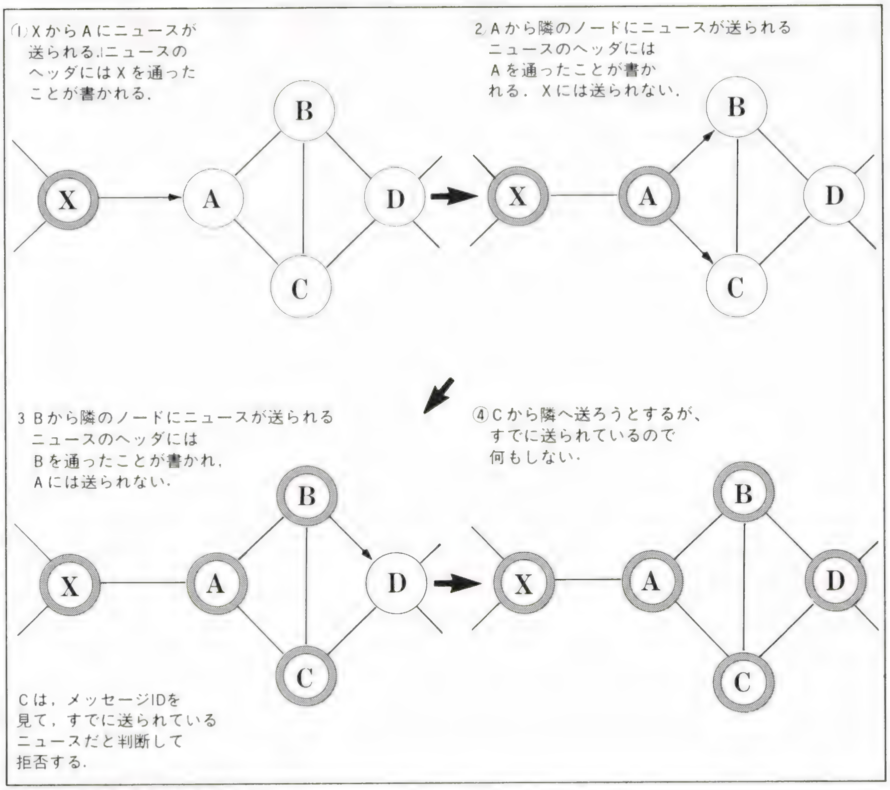


図2　無駄な転送を防ぐ仕組み

■ニュースシステムの仕組み

実際にどのようにして，このニュースシステムの仕組みが実現されているのか簡単に説明しましょう．

前回説明したように，世界中にはいろいろなネットワークがあり，ネットワーク同士が相互に接続されてさらに大きなネットワークを構成しています．

誰かがあるコンピュータから記事（ニュース）をポストしたとしましょう．するとネットワークにつながっているコンピュータは，それぞれ自分がニュースを伝えなければならない隣のコンピュータを知っていて，お隣にあたるコンピュータすべてにそのニュースを伝えるのです．そしてニュースを受け取った隣のコンピュータは，さらに自分の隣へとニュースを伝えます．こうして次々にニュースが転送されていって世界中にばらまかれるのです．このシステムのことを“放送型(ブロードキャスト型)”といいます．

さて，ニュースはいろいろなところからポストされ，転送される経路もさまざまです．ですから，すでにあるニュースを受け取っているところに，別のところから同じものが送られてくるということもありえます．これをいちいち受け取っていては無駄ですし，場合によっては無限ループになってしまうでしょう．

ネットワーク上のニュースシステムでは，無駄な転送を防ぐために，転送される1つ1つのニュースが，自分がどの道をたどっていたかという履歴や自分のMessage-IDを持っていて，すでに送られているところが分かるようになっています．この仕組みが電子ニュースシステムのミソなのです(図2)．この仕組みを持った電子ニュースシステムのことを“Bnews”といいます．これは米国のネットワークUSENETで生まれたものですが，このBnewsというシステムが，今皆さんが使っているニュースシステムなのです．

USENETというのは，アメリカのベル研究所を中心にして，UUCPという電話回線を利用したUNIXの通信機能を使っ

て作られた電子ニュースネットワークの名前です．長い間ボランティアによって運営されてきましたが，大規模になりすぎたため，現在は専門の管理者を置いて運用が行なわれています．また，使用する通信機能も専用線やローカルエリアネットワークを用いて運用できるようになり，世界最大規模の電子ニュースシステムとなっています．

ニュースシステムの活用

ニュースシステムはいろいろな使われ方をします.「こういうことについてどう思いますか」と議論するというようなときもあれば,「こんなソフトウェアができたので使ってみてください」なんていうソフトウェアの配布に使われたり,「分からないことがあるので，ちょっと教えていただけませんか」ということに使われることもあります．

あるニュースグループに,「このことが分からないんだけど」とポストしたら，世界中の人が見ていて，その中には質問に答えてくれる親切な人が必ず1人や2人はいるでしょう．こうして，ネットワークによって世界中でお互いに助け合っていろいろな知識を共有していく重要な基盤になるのが，この電子ニュースシステムだということになります．

よく耳にする質問で,「このニュースは何人くらいの人が見てますか」というのがありますが，正確なところは分かりません．sfcのニュースグループであれば，SFCのキャンパスには学生と先生を合わせて千数百人いますが，そのトピックスに全員が興味を持っているわけではないでしょうから，多くて200～500人くらいでしょうか．

fjのニュースグループだと，約400の組織がJUNETにつながっていますから,あるニュースグループを読んでいる人が1組織あたり平均10人だとして，日本全体では約4000人が読んでいるという計算になります．ただ，このfjは現在では世界中にも流しています．

では，compやaltなどの国際的なニュースグループではどのくらいの人が読んでいるでしょうか．1985年頃のことですが，約25万人が読んでいるんじゃないかという話がありました．もちろんアクティブな，非常に人気のあるニュースグループの話です．25万人はオーバーかもしれないけれども，少なくとも10万人が読んでいるニュースグループというのは，十分あると思います．

これはかなりのメディア力だといえます．そこで，これをいろいろなことに使おうと考える人も出てきます．たとえば，政治活動や商品のコマーシャルなどといったことにです．しかし，大半のネットワークはボランティアで成り立っているもので，商業ベースのものではありません．そういう商業目的の使い方はしてはいけないということが決められています．一般に，大学が加入しているネットワークを使って直接営利活動をしてはいけないのです．皆さんもこのことにはよく注意してください．

■ニュースシステムの形態

では，具体的にこのニュースシステムをどう使うかということをお話ししましょう．前回の電子メールの講義のとき，郵便局のコンピュータがあるという話をしましたね．そこに皆さん宛てのメールはためられており，これを自分のコンピュータへ持ってきて読むという形になっています．これと同じように，このキャンパスネットワークにはニュースを管理するコンピュータがあります．そこにいろいろなところからきたニュースがためておかれ，ここから皆さんは自分の使っているコンピュータにニュースを持ってくるという仕組みになっています．

しかし，電子メールのシステムとはいくつか違ったところがあります．まず，自分が1度読んだニュースは「これはすでに読みましたよ」というマークが付いて，次にニュースを読むときには表示されなくなります．この記録は自分のホームディレクトリの下の‘.newsrc’というファイルに保存されます．.newsrcには自分が読むことのできるニュースグループが登録されています．

でも，前に読んだニュースをもう1度見たいな，というときもありますね．これはある程度はできます．「ある程度」というのは，実際の掲示板でも日にちが経つと古い掲示からはずされていきますね．この電子ニュースでも世界中の人がニュースをポストしていくわけですから，毎日ものすごい数のニュースが増えていきます．それらはニュースを管理するコンピュータにためておかれるわけですが，放っておくとそのコンピュータのディスクがパンクしてしまいます．ですから，古いものから順に消していくわけです．この古いものを消すことを“expire”といいます．どのくらいで消されるか(expiration period)はシステムによって違いますし，ニュースグループによって変えたりします．これらはネットワークの管理をしている人が決めます．

しかし,「この記事はどうしても重要なので1カ月は貼っておいてくれ」といったことも指定できるようになっています．ちなみにSFCのexpiration rateは，fjのニュースが1カ月，sfcとkeioのニュースが4カ月となっています．

1度見たものは原則として見えなくなりますが，ディスク上からその記事が即座に消されてしまったというわけではないのです．

ニュースシステムを実際に使う

それでは実際にニュースシステムを使ってみることにしましょう．電子メールを読み書きするときは‘mh-rmail’というEmacsのプログラムを使いました．ニュースを読み書きするときには‘gnus’というプログラムを使います．gnusは，九州工業大学の梅田さんが作ったものです．

■gnusの使い方

では，gnusを起動してみましょう．Emacsを立ち上げてから‘Meta-x gnus’とやるか，シェルから，

```
% emacs -e gnus↵
```

を実行してください．オープニングの画面が表示されて，ニュースグループの一覧が表示されるまで少し時間がかかりますが，これはニュースを管理するコンピュータからニュースのリストを持ってくる作業をしているのです．

最初にニュースグループのリストが出てきます(**図3**)．一番左側の数字は自分がまだ読んでいない記事の数です．カーソルがある位置が今選んでいるニュースグループになりますが，ここでnまたは，Ctrl-nを押すとカーソルが下がり，pまたは，Ctrl-pと押すと上がります．これでどのニュースグループを見るかを選ぶことができるようになっています．

次に，選択したニュースグループを実際に見るためにはスペースバーを押します．gnusではスペースをどんどん押していけば，記事を端から順番に見ていけるような仕組みになっています．スペースバーを押すと，そのニュースグループの中の記事のリストが上のウィンドウに表示され，下のウィンドウには上のウィンドウのカーソル位置のニュースが表示されます(**図4**)．上のウィンドウのカーソルのある行には左端に'D'と出ていますが，これはもう読んだから，消していいよというマークです．消すといっても先ほど説明したように本当に掲示板から消してしまうのではなくて，「もう読んだから次回からは出さなくていいよ」という意味です．これを本当は読んだのに「まだ読んでないよ」という状態にしたい場合は，'u'を打つことによって読んでないことにすることもできます．

その次に書いてある数字はこのニュースグループの中での記事の通し番号です．それから，[ ]の中の数字はその記事の行数です．コロンの次は，ポストした人のログイン名がこの [ ] の中に入るだけ書いてあります．それから最後に記事のSubjectが表示されています．

■ニュースのフォーマット

次に，下のウィンドウに表示された記事を見てください．1個1個の記事は電子メールとまったく同じ形，つまりRFC822フォーマットで書かれています．上のほうにはヘッダがありますね．まず最初の行に'From: 'とあって，ここで誰がこの記事をポストしたかが分かります．次の行の'Newsgroups: 'は，この記事がどのニュースグループにポストされたのかが書いてあります．ここに複数のニュースグループを書けば，それぞれのニュースグループに対して，同じ記事を投稿することができるわけです．これを“クロスポスト”といいます．'Subject: 'には電子メールと同じで，簡単なタイトルを書きます．'Date: 'は，投稿された日付です．

次の'Distribution: 'ですが，これはどの範囲までこの記事は届くかということを制限しています．たとえば，ここにsfcと書いておけば，SFCの中だけにニュースを分配するのです．現在はsfcのニュースグループに出せば，普通はsfcの中だけに分配する，つまりDistributionは自動的にsfcになるという仕組みになっています．最後の'Organization: 'は，ポストした人の所属する組織の名前が書いてあります．このgnusでニュースを読むときには表示されませんが，ヘッダにはこのほかにも'Path: '，'Massage-ID: '，'Sender: '，'Lines: 'といった項目があります．

さて，ヘッダの下が記事そのものの内容です．全部が表示しきれていないかもしれませんが，そのときにはスペースバーをたたいてください．スクロールしてその続きの部分を見ることができます．前に戻るにはDELキーを押します．こうして，スペースバーを押しながら，最後まで見て，さらにスペースバーを押すと，今度は同じニュースグループの次の記事を表示します．また，p，nを押せば，それぞれ今見ている記事より前または次のまだ読んでいない記事を表示します．それからCtrl-p, Ctrl-nを押すと，記事のリストの上でカーソルが移動するだけで，記事を読んだことにはなりません．この場合は，読みたい記事にカーソルを移動させてからスペースバーを押すとその記事が表示されます．

そして，このニュースグループはもういいから抜けよう，つまり先ほどのニュースグループのリストに戻ろうというときには，qを押します．これでまた別のニュースグループに移ることができます．

また，gnus自体を終了するときもニュースグループのリストのところでqを押します．

図3　gnusを起動した直後の画面

```
 ↓―――――――――――――――――未読の記事の数
 84: sfc.circles
 43: sfc.cns
 25: sfc.courses.jouhou1
  2: sfc.courses.misc
117: sfc.general
158: sfc.jokes
 56: sfc.misc
 36: sfc.newsgroups    ↑PまたはCtrl-pで
233:▮sfc.questions      上へ移動
848: sfc.test
 58: keio.admin        ↓nまたはCtrl-nで
 54: keio.general       下へ移動
178: keio.jokes
172: keio.mail-lists.common-lisp
 11: keio.mail-lists.elis
  4: keio.mail-lists.smalltalk
 46: keio.misc
 22: keio.questions
147: keio.test
264: fj.ai
 14: fj.announce
  2: fj.binaries.mac
 17: fj.binaries.misc
[----]--- NGNUS: List of Newsgroups     (JJJ-:GNUS Newsgroup {nntp})--Top----
```

図4　ニュースの表示画面

```
D 214:▮[  8:t90555hh@sfc.] About zoo
  215: [ 28:isi@clab.sfc.] Re: About zoo
  216: [ 15:take@alab.sco] archives on modem1 (Re: About zoo)
  218: [ 22:mori@klab.sfc] Re: archives on modem1 (Re: About zoo)
[----]--- NGNUS: About zoo       (JJJ-:GNUS Subject)--55%-----------------
From: t90555hh@sfc.keio.ac.jp (Hideki Hashimoto)
Newsgroups: sfc.questions
Subject: About zoo
Date: 27 Aug 90 05:59:52 GMT
Distribution: sfc
Organization: Keio University Shonan Fujisawa Campus.


  comp.binaries.ibm.pcで流れているFreesoftはたいていzooでアーガイブし
てありますが、手元にこれを解凍するためのツールがないので使用できません。
どなたか譲ってくれませんか？

--
   慶應義塾大学　環境情報学部1年
               橋本　秀樹(t90555hh@sfc.keio.ac.jp)


[----]--- NGNUS: sfc.questions{214} -20 more     (JJJ-:GNUS Article)--All---
```

■ニュースのポスト

では実際に記事を投稿してみましょう(図5)．ポストするときはaを押します．まず，どのニュースグループにポストするかということを決めなければなりません．もし皆さんがあるニュースグループに入っているときに‘a’を押すと，そのニュースグループにポストすることを仮定して投稿の処理を始めます．ニュースグループのリストのところでaを押した場合には，ニュースグループはどこにしますかと聞いてきます．

ではニュースグループのリストのところでaを押してください．

Are you sure you want to post to all of USENET?(y or n)

と聞いてきましたね．これは「本当に投稿するのかい？」と確かめているのだと思ってください．sfcのニュースグループではSFC以外には届きませんが，とにかくそこは‘y’と答えます．すると，記事を書くためのウィンドウが開かれて，エコー領域にNewsgroup: と出て，「ニュースグループはどうしますか？」と聞いてきます．ここではsfc.testにしておきましょう．もし複数のニュースグループにポストをしたいと思ったら，間にカンマを入れながら，いくつかのニュースグループを書くことができます．次に‘Subject: ’と出ますから，ここではtestと入力します．そして最後に‘Distribution: sfc ’と出ますがこれでいいですから，ただリターンを押します．

そうすると準備ができて，下半分のウィンドウにNewsgroup:, Subject:, Distribution:が表示され，その下に‘--text follows this line--’と書かれてカーソルが次の行で止まっています．ここから本文を書き始めますが，この--text follows this line--という行は絶対に壊さないようにしてください．もし壊してしまったら，これとまったく同じものを入力してやる必要があります．またNewsgroup:, Subject:, Distribution:の各項目をこの時点で書き直すこともできます．そしてあとは電子メールのときとまったく同じです．

①ニュースグループのリストのところで“a”を押すと、ポストするか否かの確認をとる．

```
      84: sfc.circles
      43: sfc.cns
      25: sfc.courses.jouhou1
       2: sfc.courses.misc
     116: sfc.general
     158: sfc.jokes
      55: sfc.misc
      36: sfc.newsgroups
     213: sfc.questions
     848: sfc.test
      58: keio.admin
      54: keio.general
 [----]--- NGNUS: List of Newsgroups     (JJJ-:GNUS Newsgroup {nntp})--Top--
Are you sure you want to post to all of USENET? (y or n)
```

②次にニュースグループを指定する．

```
 [----]-----NEmacs: *post-news*          (JJJ-:News Fill)--All------------
Newsgroup: sfc.test
```

③タイトルを入力

```
 [----]-----NEmacs: *post-news*          (JJJ-:News Fill)--All------------
Subject: test
```

④どこに配布するかの指定．ここではこのままリターンを入力．

```
 [----]-----NEmacs: *post-news*          (JJJ-:News Fill)--All------------
Distribution: sfc
```

⑤ニュースを書いて、“Ctrl-c Ctrl-c”を入力するとポストされる．

```
Newsgroups: sfc.test
Subject: test
Distribution: sfc
--text follows this line--

  これはSuperASCIIのtestです。



 [ aあ]--**-NEmacs: *post-news*          (JJJ-:News Fill)--All------------
```

図5　ニュースをポストする手順

```
 [----]--- NGNUS: test              (JJJ-:GNUS Subject)--All--------------------
From: t90555hh@sfc.keio.ac.jp (Hideki hashimoto)
Newsgroups: sfc.test
Subject: test
Date: 18 Sep 90 07:50:55 GMT
Distribution: sfc
Organization: Keio University Shonan Fujisawa Campus.

  これはSuperASCIIのtestです。

--
  慶應義塾大学  環境情報学部1年
                橋本  秀樹(t90555hh@sfc.keio.ac.jp)


 [ aあ]--- NGNUS: sfc.test{997} -3 more    (JJJ-:GNUS Article)--All-----------
```

図6　ポストしたニュースの確認

記事を書き終えて，実際にポストするときも電子メールのときと同じく‘Ctrl-c Ctrl-c’とたたきます．

うまくポストできたかどうかを確認してみましょう(図6)．ポストしたニュースは，いったんニュースを管理するコンピュータに蓄えられるのでしたね．ですから皆さんはそこから新しいニュースを持ってこなければなりません．gnusを起動したときにその作業が行なわれたわけですが，gnusでニュースを読んでいるときにポストされたニュースはこれでは表示されません．

そこで新しいニュースがあったら送ってくれるようにニュースを管理するコンピュータに尋ねなければなりません．このためには‘g’を押します．これでsfc.testにちゃんと投稿されてるかどうかを見ることができます．ちゃんと投稿されていたようですね．

蛇足ですが，どんなポストをするときでも，正しく自分を名乗ることが大切です．すると，名前，所属，アドレス，連絡先などを毎回同じことを書かなければ

なりません．このような情報は，ホームディレクトリの‘.signature’という名前のファイルに書き込んでおけば自動的に記事の最後に付加されるようになります．

■フォローアップとリプライ

さて，今度は投稿されていた記事を読んだときに，皆さんが何をするかということです．ある記事を見たときに，「あ，その質問の答えは私が知っているから答えてあげよう」とか，「いや，それはこうなんじゃないか」と議論を続けようというときには，2通り，正確にいうと3通りの方法があります．

第1番目の方法は無視する，つまり読み捨てるということです．

第2番目は，これが一番普通の方法ですが，その記事に対して「それはこうすればいいよ」とか「私はこう思います」といった内容の記事を同じニュースグループに投稿することです．これを“フォローアップ（follow up）”といいます．こうすれば，そのニュースグループをアクセスする人全員が見ることができるわけです．

たとえば，それが非常に単純な質問で世の中の人はほとんど知っているようなことだったとき，それに対するフォローアップをしても，ほかのほとんどの人には無駄な情報だということになるかもしれません．ほかにもすごくたくさんの答えが予想されるような場合や，「秘密の答えを私にだけこっそり教えてちょうだいね」といった場合もあることでしょう．

こういったときに使うのが，投稿者に対して個人的にメールで答えるという第3番目の方法です．これを“リプライ（reply）”といいます．ほかにも，商品名がどうしても出てきてしまって宣伝活動になってしまうからニュースにはポストできないと思うときなどにもリプライはよく使われます．

さて，このそれぞれをやってみることにしましょう．まずフォローアップは，記事を読んでいて，これに対して私も何かいいたいことがあると思ったら，そこで‘f’と打ちます．そうすると，ポストするときと同じように，‘Are you …’と聞かれますが，あとの項目は自動的に設定されます．このときのSubjectは，フォローする記事のSubjectの前に‘Re:（Referenceの略）’と付いたもの，たとえば元の記事が‘Question’となっていたなら‘Re: Question’となります．あとはポストするときとまったく同じ手順でフォローアップのニュースをポストできます（図7）．

次に，ある記事に対してリプライしたいときは，その記事が表示されている状態で‘r’を押します．すると，ヘッダが自動的に書かれ，すぐにメールを書くことができます．あとは普通にメールを出すときと同じです（図8）．

あと，読んだ記事をファイルにしまっておく必要がある場合がありますね．そのようなときには‘o’と打つと任意のファイルに格納することができます．

これで大体ニュースシステムは扱えるでしょう．このような掲示板の仕組みを使って，みんなで情報交換をしていくというのがニュースシステムです．これは世界的なネットワークを活用していくうえでたいへん重要な役割を担っています．また普段の生活でも，sfcのニュースグループを使ってサークル活動の連絡だとか，グループの打ち合わせや，いろいろな質問とか，アンケートなどに効率よく使うことができるわけです．

1つ注意してほしいのは，皆さんがポストしたニュースは多くの人が見ているということです．国際的なニュースグループに投稿すると，世界中の人が読むことになります．したがって，もしそこで何かいたずらをすると，世界中の人に，「あ，慶應の藤沢キャンパスでこういういたずらをした人がいた」ということが知れ渡ります．

1つのニュースがネットワークを渡っていくためには，いろいろな所のコンピュータやリソース，そしてネットワークを維持するためのいろいろな人の手がかかっています．特に国際的なネットワークは国際電話回線や専用回線を使って成り立っています．そんないたずらのために国際電話のたくさんの料金を払ってしまうことがないようにするためにも，いたずらはしないようにしてもらいたいと思います．したがって記事は内容をできるだけ簡潔に，必要なことがきちっと分かるようにまとめることを心がけてください．

図7　ニュースのフォローアップ

```
In-reply-to: s90222ss@sfc.keio.ac.jp's message of 22 Aug 90 00:04:47 GMT
Newsgroups: sfc.questions
Subject: Re: UNIX osiete
References: <S90222SS.90Aug22090447@c04.sfc.keio.ac.jp>
Distribution: sfc
--text follows this line--

  僕はコンピュータのまったくの初心者です。UNIXについてこれ以上やさし
  書くことはできないというくらいやさしく書かれた入門書があったら教えて
  下さい。

 それはやっぱりみんなもってるSFCローカルガイドでしょう。
█




[ aあ]--**-NEmacs: *post-news*        (JJJ-:News Fill)--All---------------
```

図8　ニュースのリプライ

```
To: s90222ss@sfc.keio.ac.jp (Syouiti Saitou)
In-reply-to: s90222ss@sfc.keio.ac.jp's message of 22 Aug 90 00:04:47 GMT
Subject: UNIX osiete
--text follows this line--

   僕はコンピュータのまったくの初心者です。UNIXについてこれ以上やさしく書
   くことはできないというくらいやさしく書かれた入門書があったら教えて下さ
   い。

  それならSuperASCIIを読みましょう。
  村井先生の”プロフェッショナルワークベンチとしてのUNIX入門”
  がいいですよ
█




[ aあ]--**-NEmacs: *mail*             (JJJ-:Mail Fill)--All--------------
```

OS/2 アーキテクチャ入門

PM編 第4回 ウィンドウの概念

宮沢 篤之

これまでの連載では，プロセスやスレッド，メモリ管理機構など，主にOS/2*カーネルが提供するさまざまな機能について解説してきた．第4回である今回からは，OS/2のGUIである プレゼンテーションマネージャ（Presentation Manager*：以降PM）に焦点を当てる．

PMとは何か

個人で購入することが可能な価格で，個人が使用することを前提として設計されたコンピュータ．これは，パーソナルコンピュータの根本に流れる思想である．

パーソナルコンピュータが実際に広く普及している現在では，現実を踏まえ，より具体的な定義が可能だ．誤解を恐れずにいえば，それは，“MS-DOSが走るコンピュータ”ということになるだろう．

パーソナルコンピュータの現在を考えるうえで，MS-DOS（以降DOS）が果たした役割は大きい．しかしこれまでの連載でも述べられていたように，ハードディスクやメモリの大容量化，アプリケーションの大規模化など，さまざまな部分でDOSの許容範囲を超えた要求が起こっている．

ユーザーインターフェイスもそのうちの1つである．キャラクタモードで表示された“C>”プロンプトに対してキーボードからコマンドをタイプするのではなく，グラフィックスモードで表示されたアイコンやメニューなどを，マウスでクリックすることによってコンピュータに命令を与える．

また，画面は“ウィンドウ”と呼ばれる領域に分割され，そのそれぞれが異なるプログラムの出力を表示している．このようなグラフィカルユーザーインターフェイス（GUI：Graphical User Interface）*1は，近年，コンピュータ技術者ばかりでなく，一般のユーザーからも注目されている．

こうした背景のもと，OS/2における標準のGUIとして開発されたのがPMである．PMは，マウスやキーボードなどの入力デバイス，ディスプレイやプリンタなどの出力デバイスを仮想化し，メニューやアイコンなど，GUIを構成するさまざまな機能をプログラムに対して提供する．PM上に作成されたプログラムは，デバイス依存を回避すると同時に，一貫したユーザーインターフェイスを容易に構築できる．

さて，これまでのカーネルを主体とした解説に続いて，「OS/2アーキテクチャ入門」の後半は，PMのアーキテクチャを解説していく．

もちろん，PMはOS/2上の環境なので，カーネルプログラムと同様，PMプログラムもOS/2カーネルのもとで動作することに変わりはない．しかし残念ながら，カーネルとPMのプログラミングモデルは大きく異なっている．そういう意味では，今回を境に気分を新たにして，連載を読み進むことをおすすめしたい．

本連載では，アプリケーションプログラマの視点に立ち，PMの基本的な概念，プログラミングインターフェイスとシステムの構造，およびPMのプログラミングモデルに関して詳しく解説していく予定である．PM編第1回にあたる今回は，PMを構成する最も基本的な概念であるウィンドウに焦点を当て，その重要ないくつ

*はIBM Corp（米国）の商標です．

*1　“グーイー”と発音する．蛇足ではあるが，これと同じ発音をする“Gooey”という単語には，“べたべたした”というような意味がある．

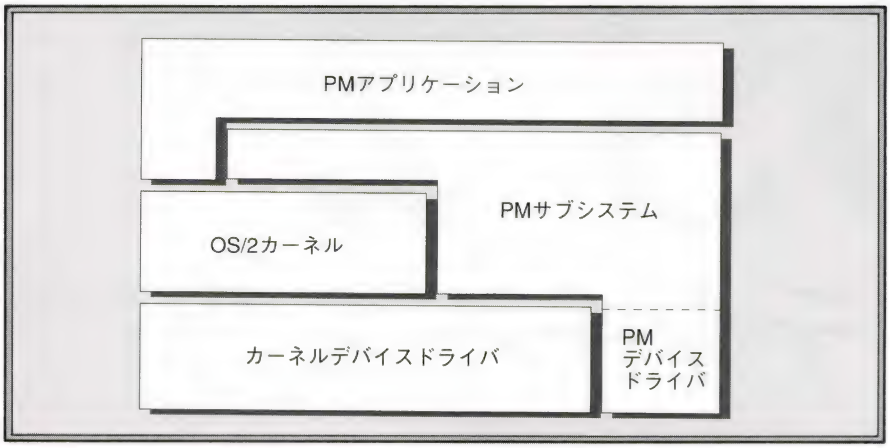


図1　OS/2カーネルとPMの関係，およびPMの構成

かの側面を述べることにしよう．

PMの設計思想

本論に入る前に，ここでPM誕生の由来と，その開発経緯について簡単に紹介しておこう．

OS/2は，80286もしくは80386CPUを搭載したパーソナルコンピュータ上における，DOSを継ぐ標準的なオペレーティングシステム(以降OS)として，Microsoft とIBMによって開発された．その第1バージョン(Ver.1.0)が発表されたのは1987年4月．このVer.1.0には，キャラクタモードのコマンド行インターフェイス(Command Line Interface)が提供されており，DOSとほぼ同じコマンド体系が備えられていた．すなわちVer.1.0では，標準のGUIはサポートされていなかったのである．

しかしこの発表の1年以上前の1986年には，OS/2における“先進的なプレゼンテーションサービス”を実現する拡張機能として，PMはすでに定義されていた．そして'86年9月，MicrosoftとIBMのハーズレー(Hursley)研究所によるPMの共同プロジェクトが実際に始動している．

このときPMは，次のような特徴を持つウィンドウシステムとして設計された．

●複数のアプリケーションからの出力を同時に見ることができる．

●OSとアプリケーション機能の両方に対する，拡張されたユーザーインターフェイスを備えている．

●PMのプログラムに対して，高度に洗練されたAPI (Application Programming Interface) を提供する．

PMアプリケーションは，システムから提供されるAPIを利用することによって，高機能なGUIを簡単に実現できなければならない．どのアプリケーションもAPIを通してプログラミングされれば，複数のアプリケーションわたって一貫したGUIが自然に構築されることになる．

このような目標のもとに定義されたPMのGUIの仕様は，Ver.2.0以降のMS-Windows (以降Windows) にも影響を与えた．そしてまたこれは，IBMのSAA*(Systems Application Architecture)*2へと引き継がれていくことになる．

PMの位置付け

このようにOS/2は，その開発当初からPMを念頭に置いて設計されている．この点がDOS＋Windowsと決定的に違う点である．OS/2カーネルとPMは，それぞれの役割に従って，論理的な切り分けがきちんとなされている．

OS/2カーネルとPMの関係，およびPMの構成を図1に示す．図から分かるとおり，PMは，次の2つの層で構成されている．

●PMサブシステム層

PMアプリケーションに対して，各種グラフィック処理やウィンドウ制御などのサービスを提供する．実際には，ひと組のダイナミックリンクライブラリになっている．

●PMデバイスドライバ層

ハードウェアに依存した処理を行なう．このようにハードウェアに依存する層が明確に切り分けられているため，上位のPMサブシステム層はハードウェアから透過になる．したがってPMサブシステム層に対する機能の追加などは，ハードウェアとはまったく独立に行なうことが可能となる．

ここでもう1度強調しておきたい．PMは完全にOS/2の一部分である．Windowsのように，後からOSに不足していた機能を補う目的で設計されたものではない．

ウィンドウとは何か

いよいよ今回の本題である“ウィンドウ”に話題を移そう．

PMに限らず，一般にウィンドウシステムのアーキテクチャについて考えるとき，まず最初につきあたるのは次のような疑問ではないだろうか．

「そもそも，ウィンドウとは一体何なのか？」

ウィンドウシステムのユーザーの視点に立ってこの疑問に答えるとするなら，「それは文書の1ページである」というのが，最も一般的な解答となるだろう．事実，'80年代初頭にXeroxから発表されたXerox Starにおいては，ユーザーに実際のオフィスの机上を連想させる，いわゆる“デスクトップメタファ”(Desktop Metaphor)によって，1つ1つのウィンドウの中に見たままの文書が表示された．

ウィンドウシステム一般ではなく，PMでいうウィンドウは具体的に何なのだろうか．ユーザーから見れば，この答えは

*2　パーソナルコンピュータからメインフレームに至るまで，IBMが提供するコンピュータラインアップにおいて，統一されたアプリケーションの開発環境，プログラムのポータビリティ，一貫したユーザーインターフェイスを実現するために定義されたソフトウェアインターフェイスおよびプロトコル．

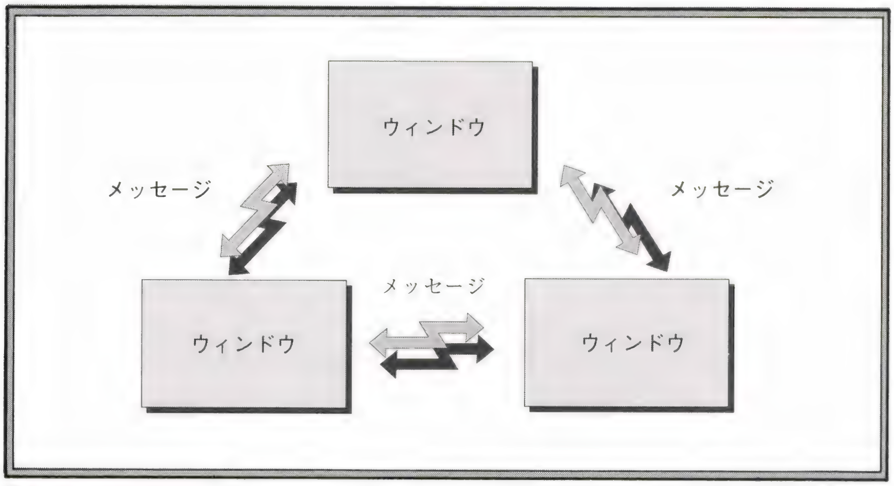


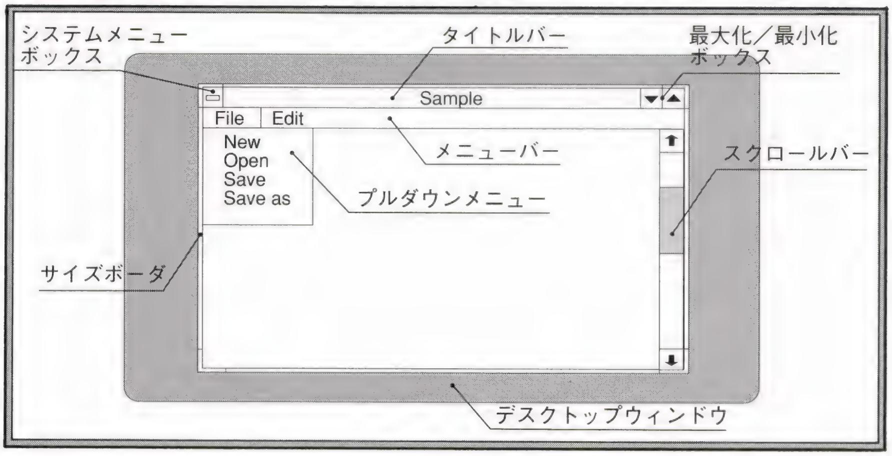
図2 ウィンドウとメッセージ

簡単に思える．それはワードプロセッサの文書の１ページであるかもしれないし，コマンドプロセッサのプロンプト画面であるかもしれない．あるいはファイルシステムが表示する，ディレクトリかもしれない．

しかしプログラマの視点に立つなら，それは何も画面に表示されるものばかりとは限らない．確かに，PMにおけるウィンドウは，画面上に一定の大きさを持った矩形領域として表示される場合が多い．そしてアプリケーションプログラムは，このウィンドウを通して，ユーザーからの入力を受け取ったり，ウィンドウに対して何らかの結果出力を行なったりする．しかし中には，プルダウンメニューなどのように，ごくたまにしか表示されないものや，あるいは決して画面上に現われることなく，他のウィンドウの状態を制御するようなものもある．PMでは，これらもまた立派なウィンドウとして，他のウィンドウと何の区別なく定義されている．

それではPMにおいて，プログラマから見たウィンドウとは何なのだろうか．それは，もっと一般的に「“メッセージ”(Message)を受け渡しする対象」だといえる．つまりウィンドウは，いわゆる“オブジェクト指向プログラミング”(Object-Oriented Programming)でいうところの“オブジェクト”(Object)であり，メッセージを受け取ってはそれに応じた“メソッド”(Method)*3を実行するものなのである．ただしここでは，PMにおけるメッセージやメソッドの詳細については述べない．これらは，連載が進むにつれて明らかになるだろう．とりあえず現段階では，たくさんのウィンドウが互いにメッセージを送りながら，ウィンドウシステム全体の処理が行なわれていく．そんな姿を思い浮かべてみていただきたい（図２）．

図3 PMにおける典型的なウィンドウ 注：サイズボーダは，実際にはフレームウィンドウの一部になっている



PMにおけるウィンドウ

PMではさまざまなウィンドウがあり，それらがメッセージをやり取りしていることは分かった．またウィンドウの一般的な定義も分かった．しかし，目に見える範囲でよいから，PMにおける具体的なウィンドウの姿を知りたいものだ．ほとんどの読者は，PMの画面をご覧になった経験をお持ちだろう．だがひょっとすると，思いもよらなかったものが，実はウィンドウになっているのかもしれない．

PMで表示される，ごく一般的なウィンドウを図３に示す．個々のウィンドウについては，引き出し線をひいてウィンドウの名前を記述してある．一見すると１つのウィンドウに見えるものが，実際には多くのウィンドウの集まりになっていることが分かる．

ひとまとまりのウィンドウを包み込んでいるのは，フレームウィンドウである．PMのもとで動作するアプリケーションは，通常，フレームウィンドウと呼ばれるウィンドウを１つ作成する．そしてフレームウィンドウ内部には，クライアントウィンドウ，プルダウンメニュー，タイトルバー，メニューバー，システムメニューボックス，最大化／最小化ボックス*4といったウィンドウが表示されている．クライアントウィンドウは，アプリケーションが入力を受け取ったり，出力を行なったりするウィンドウの作業領域である．そしてこれらのウィンドウ同士は，メッセージを使って通信し合い，共同して１つのウィンドウであるかのように振る舞うのだ．

これらのウィンドウには，親子関係がある．たとえば図３では，フレームウィンドウが親ウィンドウで，クライアントウィンドウやタイトルバーなどがその子ウィンドウにあたる．ウィンドウ同士の関係としては，このほかにも所有権(Ownership)というものもある．ウィンドウの親子関係，所有権については後で詳しく

*3 メッセージに対してどのような挙動をするかということ．

述べるが，たとえば親子関係のあるウィンドウ同士は，画面上でそれらがどのように表示されるかなど，ある規則のもとで扱われることになる．

ウィンドウを構成する要素

前述したように，PMプログラマにとってのウィンドウは，メッセージを受け渡しする対象である．しかしこれだけでは，あまりに抽象的すぎたかもしれない．そこで次は，ウィンドウが，PM内部でどのように表現されているかについて触れよう．

PM内部においてウィンドウは，次のような2つの要素によって表現される．1つは，ウィンドウの現在の状態を表わすデータ構造体である．これは，ウィンドウインスタンス構造体(Window-Instance Structure）と呼ばれる．プログラムがWinCreateWindow*5というAPIを呼び出してウィンドウを生成すると，このデータがウィンドウヒープというシステム領域に作成される．そしてWinCreateWindowは，ウィンドウハンドル（Window Handle）*6という一種の通し番号を戻り値として返す．その後プログラムからは，このウィンドウハンドルの値を使って，特定のウィンドウを識別することになる．これはちょうど，DOSにおけるファイルハンドルを思い浮かべるとよい．

プログラムが生成するウィンドウ以外にも，PM自身がシステムの初期化時に生成するウィンドウもある．これらは，デスクトップウィンドウ(Desktop Window）やデスクトップ・オブジェクトウィンドウ（Desktop Object-Window）などだ．

デスクトップウィンドウとは，PMセッションの最下部にある背景色のウィンドウである(**図3**参照)．アプリケーションによって作成され，表示されたすべてのウィンドウは，デスクトップウィンドウの子孫となる．一方，画面に表示されないすべてのオブジェクトウィンドウは，デスクトップ・オブジェクトウィンドウの子孫となる．これらのウィンドウを識別するためのウィンドウハンドルとしては，それぞれHWND_DESKTOPとHWND_OBJECTという定数があらかじめ定義されている．アプリケーションは，これらの定数をそのまま使って，どちらのウィンドウをも参照可能である*7．

ウィンドウを構成するもう1つの要素は，ウィンドウプロシージャ（Window Procedure)である．これは，ウィンドウがユーザーにどのように見えるのか，あるいはユーザーの入力に対してどのように応答するのか，といったことを記述した関数になっている．このウィンドウプロシージャについては，ここではこれ以上は触れない．ウィンドウプロシージャに関しては，次回でメッセージを解説するときに，併せてその詳細を述べる予定である．

さて，ウィンドウインスタンス構造体に話を戻そう．先に複数のウィンドウ同士が親子関係などで結ばれ，ひとまとめとなって動作することは述べた．このウィンドウ同士の関係は，PM内部では，ウィンドウインスタンス構造体を使った木構造として表現されている．ウィンドウインスタンス構造体には，次のような情報が含まれている．

- 次の兄弟ウィンドウのハンドル
- 親ウィンドウのハンドル
- 最初の子ウィンドウのハンドル
- 所有者ウィンドウのハンドル
- ウィンドウの大きさと位置
- ロックカウント
- ウィンドウスタイル
- ウィンドウ識別子
- ウィンドウクラス構造体へのポインタ
- 更新領域のハンドル
- メッセージキューハンドル
- ウィンドウプロシージャへのポインタ
- 表示パラメータへのポインタ

まだ説明されていない単語がたくさんあるので，戸惑ってしまった読者がいらっしゃるかもしれない．しかしこれらについては，折に触れて解説するので心配する必要はない．

これらの項目の値や意味は，ウィンドウマネージャによって定義されたシステムに固有のものである．しかし必要なら，アプリケーションからこれらの値を変更することが可能である*8．またウィンドウインスタンス構造体を拡張し，プログラムでその領域を自由に使用することもできる．この領域はウィンドウワード(Window Words）と呼ばれ，実際にはウィンドウインスタンス構造体とともにウィンドウヒープ上にとられ，PMのヒープマネージャによって動的に管理されることになる（**図4**）．

ウィンドウの親子関係

一般に，文書には複数の表やグラフが含まれるといった階層構造がある．初めに，ウィンドウとは文書の1ページであり，大部分のアプリケーションでそのような意味付けがなされていると述べた．もしそうであるなら，ウィンドウシステムにも階層構造を実現するような機能が必要になる．すでに簡単に触れたが，このような機能は，ウィンドウ同士が親子関係による階層構造を持つことによって実現される*9．

ウィンドウの親子関係は，主に画面上でのウィンドウの見え方を規定する．余談ながら，最近注目されているMDI(Multiple Document Interface）*10は，その応用分野として，文書処理を強く意識して

---

*4　タイトルバー，メニューバー，システムメニューボックス，最大化／最小化ボックスは，ひとまとめにしてフレームコントロールと呼ばれる．一般にPMのアプリケーションが作成するウィンドウは，フレームウィンドウ，クライアントウィンドウ，フレームコントロールの3種類に大きく分けられる．

*5　フレームウィンドウとフレームコントロールを1度に生成するには，WinCreateStdWindowというAPIを使用する．

*6　ウィンドウハンドルに対するPM特有の型宣言（hwnd）から，“エイチウィンド”と呼ばれることもある．

*7　あるいは，WinQueryDesktopWindow, WinQueryObjectWindowというAPIを呼び出すことにより，それぞれデスクトップウィンドウおよびデスクトップ・オブジェクトウィンドウの実際のウィンドウハンドル値を取得することもできる．

*8　このためのAPIとしては，WinEnableWindow（ウィンドウスタイルの変更）やWinSubclassWindow（ウィンドウプロシージャへのポインタの変更）などがある．

*9　ウィンドウの木構造（Window-Tree Structure）と呼ばれることもある．

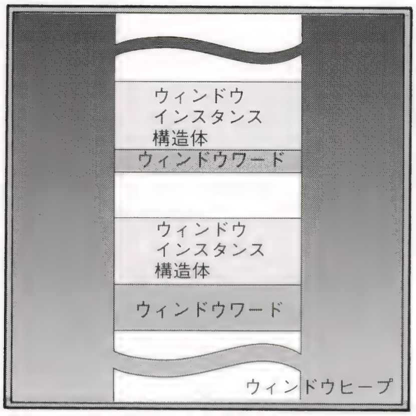


図4　ウィンドウインスタンス構造体とウィンドウワード領域

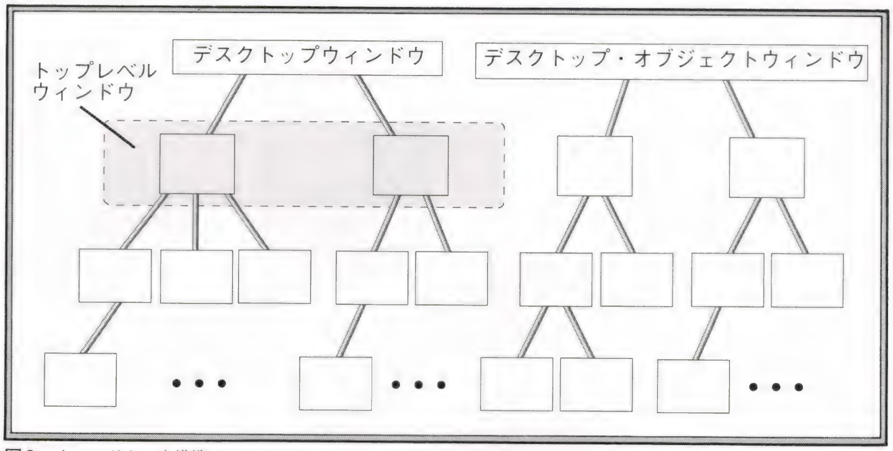


図5　ウィンドウの木構造

いることは間違いない．

ウィンドウの木構造の例を**図5**に示す．PMでは，すべてのウィンドウは，ただ1つの親ウィンドウから作成される．ただし，前述したデスクトップウィンドウとデスクトップ・オブジェクトウィンドウは例外で，これらには親ウィンドウがない*11．これは両者とも，常にウィンドウの木構造の最上位に位置するためである．

あるウィンドウに対する親ウィンドウはただ1つしかないが，子ウィンドウ(Child Window)は複数持つことが可能だ．そして子ウィンドウは，さらに複数の子ウィンドウ（親から見れば孫ウィンドウ）を持つことができる．この際，木構造の深さに関する制限はない．

デスクトップウィンドウの子は，特にトップレベルウィンドウ（Top-Level Window)と呼ばれる．通常は，このトップレベルウィンドウが個々のアプリケーションを代表することになる．

同じウィンドウを親として持つウィンドウ同士は，兄弟ウィンドウ（Sibling Window)と呼ばれる．複数の兄弟ウィンドウの間には，親ウィンドウを最下位にしたZ方向の順序（Z Order)*12が定義されている．このZ方向の順序によって，兄弟ウィンドウ同士が画面上でオーバーラップする場合の順序が決定される．Z方向の順序を変更する場合などのために，便宜上，HWND_TOPとHWND_BOTTOMというウィンドウハンドルがあらかじめ定義されている．名前からも想像できるように，HWND_TOPはZ方向の最上位にあるウィンドウを，HWND_BOTTOMは最下位にあるそれぞれ仮想的なウィンドウを指す．

ウィンドウインスタンス構造体を利用した，ウィンドウの木構造の概念図を**図6**に示す．このようにPMにおけるウィンドウの木構造では，親子関係は双方向リストとして，また兄弟関係は片方向リストとして実現されている．

ウィンドウの親子関係はまた，ウィンドウの移動，クリッピング，破壊に関する親から子への一方的な関係でもある．ウィンドウの位置は，その親ウィンドウの左下隅を原点とした相対位置で定義されている．したがって親ウィンドウが移動された場合，すべての子ウィンドウも親ウィンドウと一緒に移動される．

ウィンドウは常にその親ウィンドウによってクリップされ，親ウィンドウの境界を超えた部分は表示されない*13．そしてプログラムがWinDestroyWindowというAPIを呼び出すことによって親ウィンドウが破壊されると，そのすべての子ウィンドウもまた破壊される．この場合，まず子が先に破壊され，親は一番最後に破壊されるので，親ウィンドウは自分が管理するデータやリソースを退避したり，解放したりすることができる．

ウィンドウがどのように表示されるか，またどのように動作するのかといったことは，後で述べるウィンドウプロシージャの記述によって，アプリケーション側でかなり自由に定義できる．しかし，ウィンドウの親子関係が定める規則については，プログラムから変更することは絶対にできない．

ウィンドウスタイル

ウィンドウスタイルは，ウィンドウの動作や表示に関する属性である．具体的には，各種のスタイルビットをOR演算子で連結した，ダブルワード（32ビット）長のビットフラグの集まりになっている．このうち，上位ワードに格納される11種類のスタイルは，標準スタイルとして，システムによってあらかじめ定義されている（**表1**）．

一方，下位ワードには，個々のウィンドウクラスに特有なスタイルビット，もしくはユーザー定義可能なスタイルビットが格納される．

ここでは，ウィンドウの親子関係によって影響を受けるウィンドウスタイルとして，WS_VISIBLEとWS_DISABLED

*10　WindowsおよびPMにおいて，複数の子ウィンドウを使用するアプリケーションのユーザーインターフェイスを規定したもの．

*11　これらのウィンドウに対するウィンドウインスタンス構造体では，親ウィンドウのハンドル値としてNULLが格納されている．

*12　X方向が画面の幅であり，Y方向が画面の高さであるなら，深さはZ方向ということになる（ただしこれは，具体的な数値として与えられるわけではない）．兄弟ウィンドウ同士がオーバーラップした場合，Z方向の最上位にあるウィンドウが一番手前に表示される．

*13　通常のクリッピング処理は，自分自身のウィンドウを境界として行なわれる．ただし例外として，ウィンドウスタイルにWS_PARENTCLIPが設定されている場合には，そのウィンドウ自身の境界を超えて描画することができてしまう．しかしいずれにしても，親ウィンドウの境界を超えた部分は表示されない．

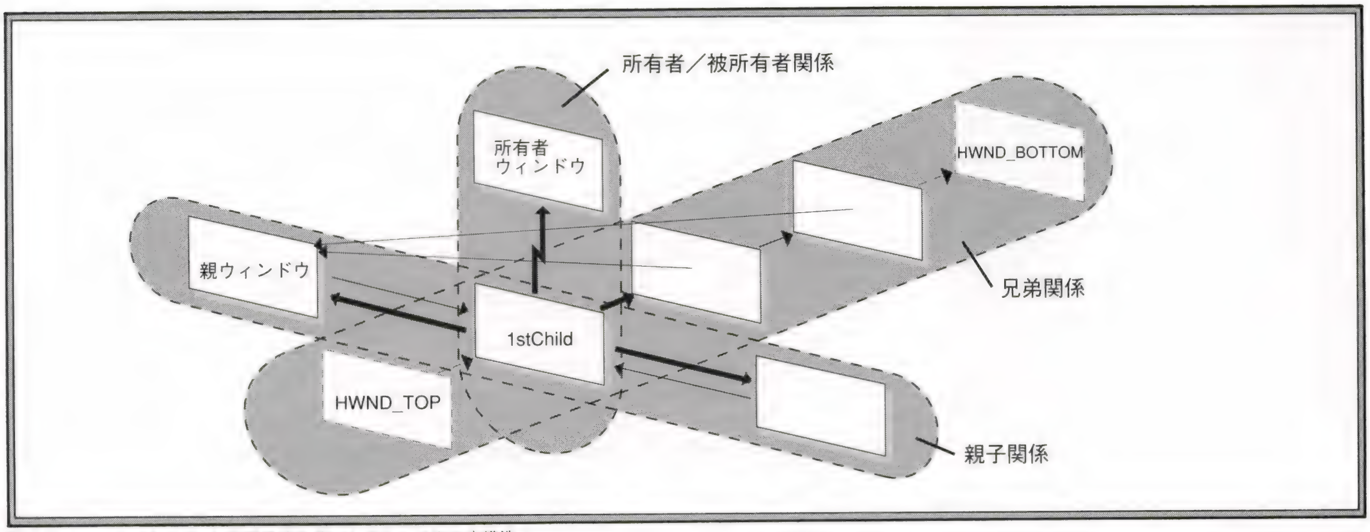


図6　ウィンドウインスタンス構造体を使ったウィンドウの木構造

を取り上げることにする．

WS_VISIBLEは，ウィンドウの可視／不可視状態を決める属性である．ウィンドウスタイルにこの属性が設定されていると，PMはそのウィンドウを画面上に描画する．ここで重要なのは，明示的にこのウィンドウスタイルを指定しない限り，生成されたウィンドウは不可視になるということである．PMばかりでなく，一般にウィンドウシステムにおいては，画面への描画やクリッピングは負荷の大きい処理である．このためPMでは，これらのスタイルが省略されたときには，なるべく描画を行なわないように，また描画する場合でも，なるべくクリッピングを行なわないようになっている*14．

親ウィンドウが不可視状態になると，その子がたとえWS_VISIBLEを持っていたとしても，子ウィンドウは不可視状態になる．ただしWinShowWindowを使えば，プログラムから明示的に可視／不可視状態を変更することができる．

あるウィンドウが現在可視状態かどうかを調べるには，WinIsWindowVisibleを用いる．このとき，状態を調べるウィンドウがオーバーラップするウィンドウによって完全に隠されている場合など，たとえ画面に表示されていなくても，WS_VISIBLEが設定されている限り可視状態だと解釈される．目的のウィンドウが現在画面上に表示されているかどうかを調べたければ，WinIsWindowShowingを利用すればよい*15．

WS_DISABLEDは，ウィンドウが一時的にキーボードやマウスからの入力を受け付けないようにするためのスタイルである．たとえば，ダイアログボックスが現われると，その親ウィンドウは使用禁止状態になる．使用禁止状態のウィンドウをクリックすると，システムはビープを鳴らしてそのウィンドウが使用禁止であることをユーザーに通知する．

この使用禁止状態も，親から子へ伝播する．親ウィンドウが使用禁止状態になると，その子がたとえWS_DISABLEDでなかったとしても，使用禁止にさせられる．この属性を変えるには，WinEnableWindowを使用する．

表1　ウィンドウスタイルビット一覧

ウィンドウスタイル識別子	意味
WS_VISIBLE	ウィンドウを可視状態にする
WS_DISABLED	ウィンドウへのマウスやキーボード入力を禁止する
WS_CLIPCHILDREN	ウィンドウの描画時，その子ウィンドウの領域は除外する
WS_CLIPSIBLINGS	ウィンドウの描画時，その兄弟ウィンドウの領域は除外する
WS_PARENTCLIP	ウィンドウの描画は，その親ウィンドウの境界によりクリップされる
WS_SAVEBITS	ウィンドウが移動されたり，可視状態になったりしたとき，その下に隠れる画面イメージを保存する
WS_SYNCPAINT	ウィンドウのある部分が無効になったら，直ちにWM_PAINTメッセージが送られる
WS_MINIMIZED	ウィンドウを最小化した状態にする
WS_MAXIMIZED	ウィンドウを最大化した状態にする
WS_GROUP	ダイアログ項目のグループ内における，最初の項目として識別する
WS_TABSTOP	タブストップウィンドウとして識別する

オブジェクトウィンドウ

気の早い読者の中には，ウィンドウの親子関係の木構造から，メニューの階層構造を連想された方がいらっしゃるかもしれない．しかし実際にPMを起動して，メニューバーをクリックし，プルダウンサブメニューを表示してみていただきたい．プルダウンサブメニュー(Pull-Down SubMenu)は，メニューバーの外側に表示されることが分かるだろう．すでに述べたとおり，子ウィンドウは，必ずその親ウィンドウによってクリッピングされる．したがってプルダウンメニューは，メニューバーの子ウィンドウではありえない．また階層化メニューにおいても，表示されるプルダウンメニューは直前のメニューの外側に表示されるので，これらにも親子関係はない．結論から先にいうと，メニューバーはフレームウィンドウの子であるのに対して，プルダウンメニューはデスクトップ・オブジェクトウィンドウの子として定義されているのである．

これもすでに述べたとおり，デスクトップ・オブジェクトウィンドウの子孫は，すべてオブジェクトウィンドウと呼ばれ

るものになる．そしてオブジェクトウィンドウは，画面に表示されることなく，ただメッセージをやり取りするためだけに生成されたウィンドウである．オブジェクトウィンドウは，たとえウィンドウスタイルとしてWS_VISIBLEを持っていたとしても，決して画面上には表示されない．しかしウィンドウの親子関係や，次に述べる所有者／被所有者関係に関する規則は，他のウィンドウとまったく同じように適用される．重要なのは，オブジェクトウィンドウは，ウィンドウスタイルのようなウィンドウの属性ではないということである．したがって親ウィンドウをデスクトップ・オブジェクトウィンドウ（もしくはその子孫）からデスクトップウィンドウ（もしくはその子孫）に変えれば，(可視状態であれば)そのウィンドウは画面に表示されることになる．

今述べたプルダウンメニューを実現している手法は，まさにこれである．ユーザーがメニューバーをクリックすると，対応するプルダウンメニューの親ウィンドウがデスクトップ・オブジェクトウィンドウからデスクトップウィンドウに変更される．こうしてプルダウンメニューは，画面に現われることとなる．

ウィンドウの所有者/被所有者関係

ウィンドウ同士のもう１つの関係として，“所有者ウィンドウ”(Owner Window)とそれに所有される“被所有者ウィンドウ”(OwneeまたはOwned Window) というものがある．この“ウィンドウの所有権”(Ownership)は，親子関係とは独立に設定することができる．ウィンドウ同士が所有者／被所有者関係を結ぶと，メッセージの経路が確保される．

たとえば，ダイアログボックスの中に表示される“Ok”ボタンは，その親であるダイアログウィンドウによって所有されている．ユーザーがこのOkボタンを選ぶと，ボタンは所有者であるダイアログウィンドウにメッセージを送る．

しかしこのウィンドウの所有者／被所有者関係について，あるいは疑問を持った読者がいらっしゃるかもしれない．「ウィンドウとは何か」で述べたように，ウィンドウとはメッセージを受け渡しする対象であり，なにも所有者／被所有者関係がなくとも，メッセージのやり取りができるはずである．また所有者／被所有者関係を結んだからといって，PMがアプリケーションに代わって適当なメッセージを送るというわけでもない．つまり所有者／被所有者関係には，親子関係のように，システムであらかじめ定められた規則はないのである．たとえ所有者／被所有者関係があるウィンドウ同士でも，実際にどのようなメッセージのやり取りを行なうかは，両者のウィンドウプロシージャで記述されなければならない．

それでは，ウィンドウの所有者／被所有者関係とは，いったい何のためにあるのだろうか．この答えを探るには，PMが備える種々の基本概念が，どのように導入されたのかを考える必要がある．

PMを形成する基本的な概念の中には，それらが導入された理由が明確でないために，その本質がなかなか理解できないものが少なくない．これがたとえば，CPUが行なう算術演算命令などであれば，曖昧さが入り込む余地はないだろう．しかしウィンドウシステムでは，その自由度が非常に大きいこと，歴史が浅いことなどから，システムの基本原理といったものが，まだ十分に確立されていないのだろうと筆者は考える．

PMにさまざまな基本概念が導入された理由としては，次のようなものが考えられるだろう．

1．必然性のあるもの
2．論理的な整合性を保つためのもの
3．歴史的な背景(既存のシステムとの互換性など) によるもの
4．設計上の誤りによるもの

たとえば，ウィンドウの作成や破壊は必然的なものであろう．一方OS/2 Ver.1.1ではあったウィンドウのロックカウントがVer.1.2からはサポートされていないところを見ると，これは，あるいは設計上の誤りによるものだったのかもしれない．

ウィンドウの所有者／被所有者関係は，第２の理由によるものだと思われる．これは，PMの前身にあたるWindowsの，ウィンドウの親子関係を整理して拡張したものなのである．Windowsでは，ウィンドウの所有者／被所有者関係はなく，親子関係しか存在しない*16．

親子関係と所有者／被所有者関係を分離したことにより，PMでは，たとえばメニューの親をフレームウィンドウに，所有者をクライアントウィンドウにすることによって，メニューコマンドが選択されたとき，メッセージが所有者であるクライアントに送られ，アプリケーションが適切に応答できるようになったのである．

●

PMは，IBMのSAAに準拠したパーソナルコンピュータ上の最初の製品であると同時に，ウィンドウのプログラミングモデルに，オブジェクト指向の概念を取り入れた最初のシステムでもある．次回は，オブジェクト指向プログラミングシステム（OOPS：Object-Oriented Programming System）としてのPMという観点から，今回特に断りなしに使ってきた，メッセージやウィンドウクラス，そしてウィンドウプロシージャなどの概念について，解説していきたいと思う．

---

*14　たとえば，そのウィンドウにWS_CLIPCHILDRENスタイルが指定されていなければ，そのウィンドウに対する描画は自分の子ウィンドウ(もしあれば)の領域をも塗りつぶしてしまう．またWS_CLIPSIBLINGSが指定されていない場合には，オーバーラップしている兄弟ウィンドウの領域も塗りつぶす．

*15　このAPIは，OS/2 Ver.1.2からサポートされた．

*16　正確には，あるウィンドウが“親ウィンドウを持つ”という関係であり，親ウィンドウを持ったからといって自分が子ウィンドウになるわけではない．子ウィンドウであるかどうかは，そのウィンドウスタイルによって決められる．ウィンドウスタイルがWS_CHILDであれば，その“親ウィンドウ”はPMでいう親と所有者を兼ねることになり，WS_POPUPであれば，単に所有者ウィンドウであるということになる．

386マシン徹底活用講座

386 de MS-DOS

連載第4回　プロテクトモードの管理

明石 真平

前回はプロテクトモードを使うための最低限の機能について説明した．今回は割り込み処理やGDT，LDT，保護機能，ゲートなど，DOS-Extenderを作るために必要な事項について解説していこう．

システムセグメント

386のセグメントディスクリプタとして，すでにコードセグメントディスクリプタとデータセグメントディスクリプタの2つについて解説したが，386にはこのほかにシステムセグメントと呼ばれる特殊セグメントがある（これに対し，コード／データセグメントはアプリケーションセグメントと呼ばれる）．

システムセグメントはプロテクトモードで稼働するプログラムを管理するために使用され，**表1**に示す12種類がある．

システムセグメントに対するディスクリプタは，アプリケーションセグメントの場合と同様に，グローバルディスクリプタテーブル（GDT）もしくはローカルディスクリプタテーブル（LDT）に格納される．各ディスクリプタのサイズもやはり8bytesであり，**図1**のような形式になっている（詳細は各ディスクリプタごとに異なる）．アプリケーションセグメントと異なるのは，ビット44が0かどうかである．typeフィールドには**表1**に示したディスクリプタタイプをセットする．

ディスクリプタタイプに286と386の2つがあるが，両者には特に機能上の違いはない．286の場合はビット56から63の内容が無視され，オフセット部は最大16Mbytesに制限されている．

それでは以下順に各ディスクリプタについて簡単に説明しておこう．

・タスクステートセグメント（TSS）

TSSは，タスクごとに固有な情報を格納するのに使用される．タスクとはMS-DOSやUNIXのプロセスに相当し，TSSには現在実行しているプログラムで使用している全汎用レジスタやセグメントレジスタ，スタックポインタ，命令ポインタなどの情報が格納される．

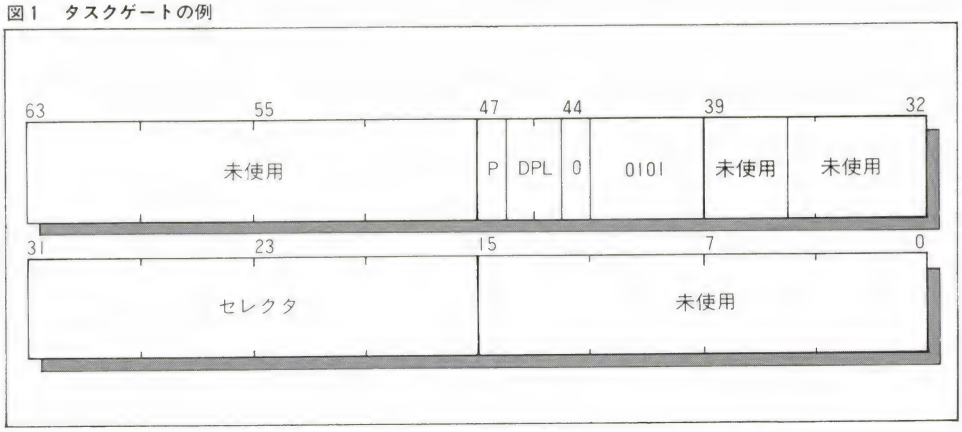
マルチタスクのOSや環境を作成する場合，TSSを使わずに独自のタスク用データ構造を用意しても構わない．実際386のUNIXやDOS-Extenderでは，TSSのデータ構造とOSの管理しているデータ構造が異なるという理由で，使用していないものも多い．しかしTSSを使用すると，タスクスイッチ（全レジスタの退避／復帰，新タスクへのジャンプなど）はCPUが自動的にしかも高速に行なってくれるので，ハードウェア割り込みの処理など，高速性を要求されるシステムには十分に使う価値がある．

TSSには「使用可能TSS(Available TSS)」と「使用中TSS（Busy TSS)」

表1　システムセグメント番号

0	予約
1	使用可能286TSS
2	LDT
3	使用中286TSS
4	286コールゲート
5	タスクゲート
6	286割り込みゲート
7	286トラップゲート
8	予約
9	使用可能386TSS
10	予約
11	使用中386TSS
12	386コールゲート
13	予約
14	386割り込みゲート
15	386トラップゲート

図1　タスクゲートの例



の2つがある．システム中ではただ1つのTSSのみが「使用中TSS」になり，その他のものは「使用可能TSS」として待機状態になる．

・コールゲート（Call Gate）

コールゲートはOSのシステムサービスを呼び出すために使用する．

・タスクゲート（Task Gate）

タスクスイッチのために使用する．

・割り込みゲート（Interrupt Gate）

このゲートと，次のトラップゲートは割り込みディスクリプタテーブル(IDT)にのみ登録することができる．

割り込みゲートには，割り込みが発生したときに起動する処理ルーチンのアドレス(セグメントセレクタ：オフセット)を記述する．

・トラップゲート（Trap Gate）

トラップゲートは割り込みゲートとほぼ同じ働きをするが，割り込みマスクフラグ（IF）の扱いだけが異なる．割り込みが発生すると，割り込みゲートでは割り込みを禁止状態にするが，トラップゲートの場合にはIFの状態はそのまま維持される．

386の保護機構について

386ではプログラムに対する保護機構として，基本的には4段階の特権レベルとセグメントをベースにした，種々のアクセス制限機構を用意している．

特権レベルは0から3まであり，0ならすべての操作が可能だが，3だと使用できる命令やセグメントへのアクセスなどに最大限の制限が課せられる．通常は0をOSのカーネルが使用し，3をユーザープログラムで使用することになる．中間のレベルは特に使わなくても構わないが，デバイスドライバや，ややユーザー寄りのサービスルーチンなどに使えるだろう（**図2**）．

CPUの現在の特権レベルCPL（Current Privilege Level）はコードセグメントのディスクリプタ中のDPL（Descriptor Privilege Field）で決まる．通常，この値はCSセグメントセレクタの最下位2ビットと同じにしておく．

図2　4段階の保護レベル例

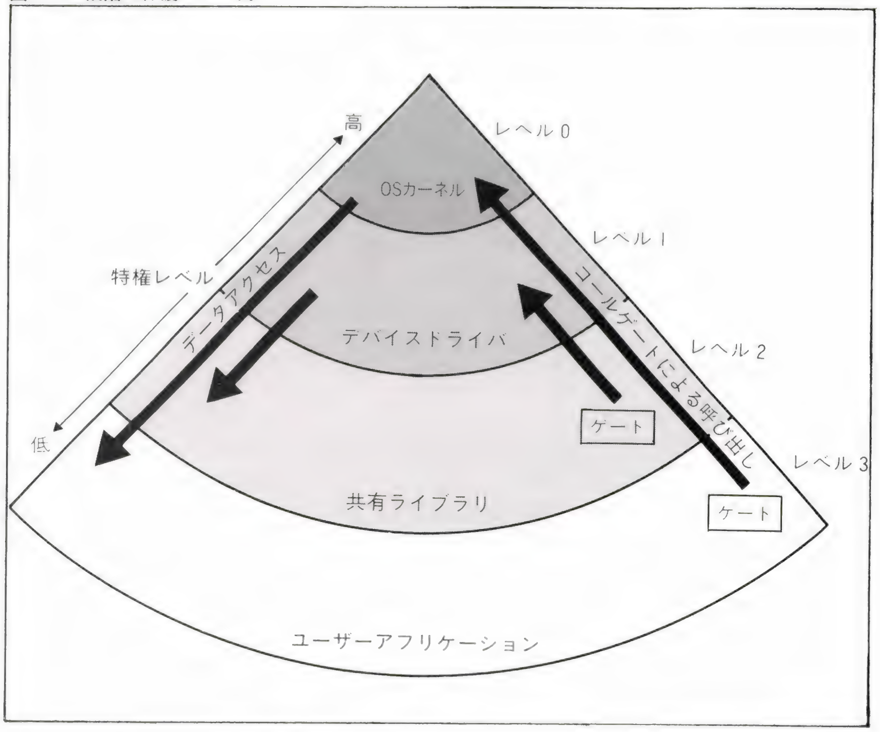
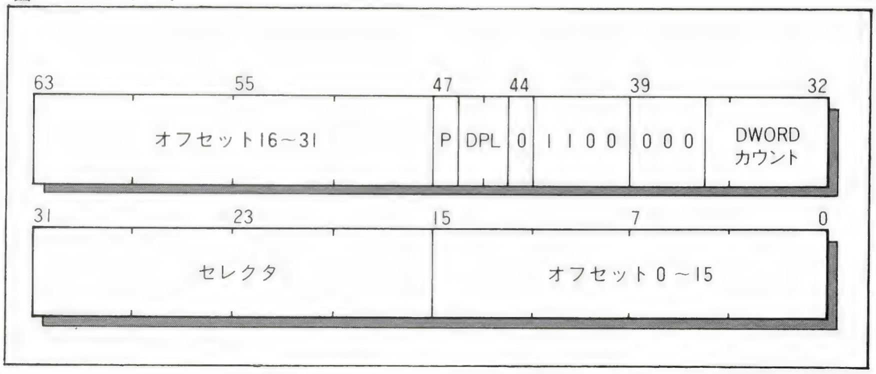


図3　コールゲート



データをアクセスする場合は，CPLとアクセス対象のデータセグメントのDPLとが比較され，CPL≦DPLでなければならない．特権レベルの高いセグメントから低いレベルのセグメントの内容をアクセスすることができるが，逆にユーザープログラムからOSの内部コードやデータをアクセスしようとすると，保護違反の例外が起こる．このチェックはセグメントレジスタにセレクタ値をロードしようとした時点(例：MOV DS,AXとしたとき）で行なわれる．

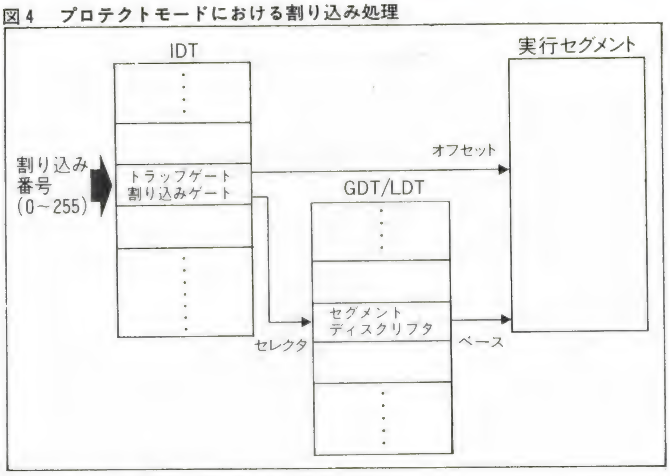
JMP farやCALL farなど，実行するコードのセグメントを変更する場合はCPL＝DPLでなければならない．つまり同じレベルのコードしか呼び出すことはできない．これは，特権レベルの高いセグメントのルーチンを簡単に呼び出せるようだと，システムが破壊される危険性が非常に大きいためである．

OSのサービスルーチンを呼び出すような場合には，ゲート（Gate）を使ってコードセグメントの特権レベルを変更することになっている．

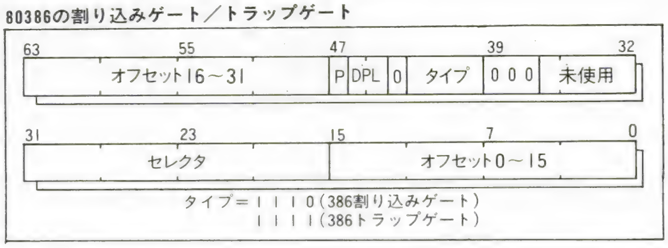
ゲートについて

ゲートはGDTやLDT中にあり，形としては通常のCALL far命令で呼び出すこと

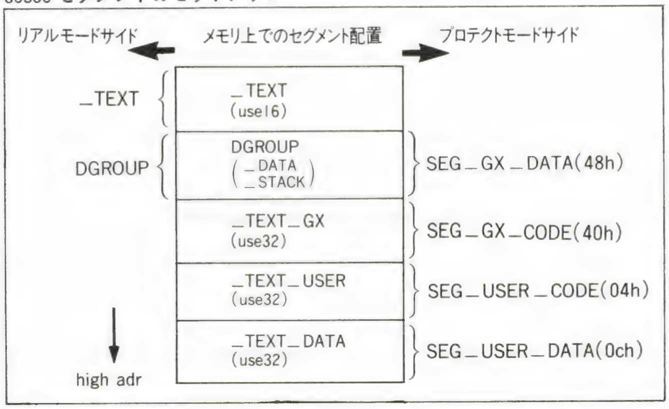
図 4　プロテクトモードにおける割り込み処理



80386の割り込みゲート／トラップゲート



80386 セグメントのセットアップ



サンプルリスト

```
                name    GX
                .386p
;**** G-Extender ver 0.0.1 ****

;=== SYSTEM SELECTION ===
_SYSTEM_PC9801 =      0        ; NEC PC9801 series
_SYSTEM_PCAT =        1        ; IBM-PC/AT compatible arch.

;=== SEGMENT DEFINITIONS ===
;** 16-bit real mode code **
_TEXT           segment para public use16 'CODE'
_TEXT           ends
;** GX data part **
_DATA           segment para public use16 'DATA'
_DATA           ends
STACK           segment para stack use16
STACK           ends
;** data is shared by both mode **
DGROUP          group   _DATA,STACK
;** 32-bit protect mode code **
_TEXT_GX        segment para public use32
TextGxOffset0   label   byte
_TEXT_GX        ends
;** user's address space (use32) **
_TEXT_USER      segment para public use32
_TEXT_USER      ends
_DATA_USER      segment para public use32
_DATA_USER      ends

;** 80386/486 segment descriptor **
_DESC           struc                   ; descriptor skelton
_LIM_0_15       dw      0               ; segment limit
_BAS_0_15       dw      0               ; base (0 ~ 15)
_BAS_16_23      db      0               ; base (16 ~ 23)
_ACCESS         db      0               ; access right byte
_GRAN           db      0               ; granurity
_BAS_24_31      db      0               ; base (24 ~ 31)
_DESC           ends

;=== segment selector definition ===
SEG_ALIAS       =       08h                     ; alias selector offset
;** GDT section **
SEG_NULL        =       00h                     ; null selector
SEG_GDT         =       08h                     ; GDT
SEG_LDT         =       10h                     ; LDT
SEG_LDT_AL      =       SEG_LDT+SEG_ALIAS
SEG_IDT         =       20h                     ; IDT
SEG_FLAT        =       28h                     ; physical memory (4GB flat)
SEG_64K_CODE    =       30h                     ; my use16 code (_TEXT)
SEG_64K_DATA    =       38h                     ; my use16 data (DGROUP)
SEG_GX_CODE     =       40h                     ; my use32 code (_TEXT_GX)
SEG_GX_DATA     =       48h                     ; my use32 data (DGROUP)
;** LDT section -- local per task **
```

になる．たとえばGDTの0080hにコールゲートがあるとすると，CALL 0080h:0でこのゲートを呼び出す（オフセット部は無視される）．ただし割り込みゲートとトラップゲートはIDT中に置かれているので，INT xxh命令で呼び出す．

コールゲートは**図 3**のようなフォーマットになっていて，通常，OSのサービスルーチンを呼び出すために使用される．セグメントセレクタは実行するべきコードのセグメントセレクタ値で，オフセットはサービスルーチンの開始オフセットを表わす．DWORDカウントにはその手続きに渡すべき引数の数をDWORD単位で数えたもので，最大31個のDWORD引数を渡すことができる．引数は，スタックを介して渡される．CPLとゲートディスクリプタ中のDPLが異なる場合，スタックがDPLのレベルのものに切り替えられ，指定された個数の引数が新しいスタックにコピーされる．

タスクゲートはタスクスイッチを行なうためのものだが，詳細はいずれ必要になった時点で触れることにする．

割り込み処理について

ハードウェア割り込みやソフトウェア割り込み，例外などが発生すると，CPUはIDTを参照して処理ルーチンを起動する．リアルモードの場合には，IDTには単に処理ルーチンのセグメントとオフセットの対が置かれているだけであったが，プロテクトモードの場合は，IDTには割り込みゲート，トラップゲート，タスクゲートのいずれかのゲートが置かれている（**図 4**）．

タスクゲートの場合は，割り込み処理の場合も通常のタスクゲートの場合と同様にタスクスイッチが発生する．しかし，用意されたゲートが割り込みゲートかトラップゲートのいずれかであった場合は，ディスクリプタ中に設定されている“セグメントセレクタ：オフセット”の指すアドレスに制御が移行する．

```
SEG_USER_CODE =      04h              ; user code space
SEG_USER_DATA =      0Ch              ; user data space
SEG_SYSCALL   =      14h              ; call gate to system-call

_SystemCallGate macro                 ;; call-gate to SEG_SYSCALL
                db    0eah, 0, 0, 0, 0 ;; jmp far SEG_SYSCALL:0
                dw    SEG_SYSCALL
                endm

_DATA           segment
;** GDT **
GdtBase         label  byte
GdtDescNull     _DESC  <>                          ; null segment
GdtDescGdt      _DESC  <_GDT_SIZE-1,?,?,92h,40h,?>  ; GDT itself
GdtDescLdt      _DESC  <_LDT_SIZE-1,?,?,82h,000,?>  ; LDT
GdtDescLdtAL    _DESC  <_LDT_SIZE-1,?,?,92h,40h,?>  ; LDT alias
GdtDescIdt      _DESC  <_IDT_SIZE-1,?,?,92h,40h,?>  ; IDT alias
GdtDescFlat     _DESC  <0ffffh,?,?,92h,0cfh,?>      ; 4GB flat data segment
GdtDesc1MCode   _DESC  <0ffffh,?,?,9ah,000h,?>      ; 64KB flat code segment
GdtDesc1MData   _DESC  <0ffffh,?,?,92h,000h,?>      ; 64KB flat data segment
GdtDescGxCode   _DESC  <0ffffh,?,?,9ah,0cfh,?>      ; kernel code segment
GdtDescGxData   _DESC  <0ffffh,?,?,92h,0cfh,?>      ; kernel data segment
_GDT_SIZE       =      ($ - GdtBase)
;** LDT **
                align  16
LdtBase         label  byte
LdtDescUserCode _DESC  <0ffffh,?,?,9ah,0cfh,?>      ; user's code space
LdtDescUserData _DESC  <0ffffh,?,?,92h,0cfh,?>      ; user's data space
LdtDescSysCall  _DESC  <SystemCall-TextGxOffset0,SEG_GX_CODE,0,8ch,0,0>
                                                    ; 386 call-gate for system call
_LDT_SIZE       =      ($ - LdtBase)
;** IDT **
                align  16
_IdtEntryMacro  macro  n
                _DESC  <IdtEntry&n&-TextGxOffset0,SEG_GX_CODE,0,8eh,0,0>
                endm
IdtBase         label  byte
_IdtEntryNumber =      0
        rept    256
                _IdtEntryMacro  %_IdtEntryNumber
_IdtEntryNumber =      _IdtEntryNumber + 1
        endm
_IDT_SIZE       =      ($ - IdtBase)
                align  4
GdtPtr          label  fword          ; GDT base & limit
GdtPtrLimit     dw     _GDT_SIZE-1
GdtPtrBase      dd     GdtBase
IdtPtr          label  fword          ; protect mode IDT
IdtPtrLimit     dw     _IDT_SIZE-1
IdtPtrBase      dd     0h
IdtRealPtr      label  fword          ; real mode IDT
IdtRealPtrLimit dw     4*256-1
IdtRealPtrBase  dd     0h
```

```
;** segment register save area, used by kernel **
RealModeSegDS   dw     ?        ; initial values are set by init routine
RealModeSegES   dw     ?
RealModeSegSS   dw     ?
; protect mode save area
ProtModeSegDS   dw     SEG_GX_DATA  ; default segment selector
ProtModeSegES   dw     SEG_GX_DATA
ProtModeSegSS   dw     SEG_GX_DATA
;
EndStr          db     'GX␣End....', 0dh, 0ah, '$'
_DATA           ends

STACK           segment
StackBottom     label  byte
                db     4096 dup (?)
StackTop        dd     ?
STACK           ends

                assume cs:_TEXT,ds:DGROUP,ss:DGROUP
_TEXT           segment
;** MACRO _GotoProtMode <dst>
;** desc : Enter protect mode and goto <dst>
;**          CS := _TEXT_GX, DS, ES, SS := DGROUP
;**          SS:(E)SP points to same addr before and after mode switching
_GotoProtMode   macro  dst
                push   word ptr SEG_GX_CODE ;; simulate 16-bit mode far call
                push   word ptr (offset dst - offset TextGxOffset0)
                jmp    near ptr GotoProtMode
                endm
;** actual mode switching code **
                align  4
GotoProtMode    proc   far
                mov    RealModeSegDS,ds ; save real mode segment
                mov    RealModeSegES,es
                mov    RealModeSegSS,ss
                push   eax              ; save eax
                db     66h              ; 32-bit form LGDT inst
                lgdt   GdtPtr
                db     066h             ; 32-bit form LIDT inst
                lidt   IdtPtr
                mov    eax,cr0          ; protection enable
                or     al,1             ; set bit-0 (PE)
                mov    cr0,eax
                jmp    short GotoProtMode1   ; flush instruction queue
                align  4
GotoProtMode1:  mov    ax,SEG_LDT       ; setup LDT
                lldt   ax
                mov    ds,ProtModeSegDS ; setup protect mode segment
                mov    es,ProtModeSegES
                mov    ss,ProtModeSegSS
                pop    eax
                ret                     ; 16bit-mode far return
```

DOS-Extenderの作成

今回のサンプルプログラムは前回より一歩進んで，より積極的にプロテクトモードを使ったものになっている．CPUをプロテクトモードにしたあと，ユーザーのプログラムをUSE32セグメントで実行する，DOS-Extenderのプロトタイプになっている．割り込みも正しく処理し，コールゲートも用意してみた．少し長くなったが，やっていることは簡単なことばかりである．

プログラムでは**図5**のようなセグメントをセットアップしている．プログラムはMASMでアセンブルするEXEモデルになっている．

リアルモードでは_TEXTとDGROUP(_DATA，_STACKを含む)の2つのセグメントを使用し，プロテクトモードでは_TEXT_GXにコードを（これに相当するセグメントセレクタはSEG_GX_CODE，以下SEG_＊はリスト中に示したセグメントセレクタ名を指す)，DGROUP(SEG_GX_DATA)にデータ，スタックを入れることにする．なお，このDGROUPはリアルモードとプロテクトモードで共有している．

この他のセグメントとしては，GDT自身を指すSEG_GDT，LDTを指すSEG_LDT，IDTを指すSEG_IDT，リアルモードのプログラム（_TEXT）とデータ（DGROUP）を指しているSEG_64K_CODE，SEG_64K_DATAなどを用意した．LDTに関してはエイリアスを表わすSEG_LDT_ALも用意した．プロテクトモードからLDTの内容をアクセス／操作する場合はSEG_LDTではなく，このSEG_LDT_ALを使用する．

LDT内にはユーザーのプログラムコードとデータセグメントを表わすSEG_USER_CODE，SEG_USER_DATAを用意した．さらにコールゲートとしてSEG_SYSCALLも用意している．システムコールを行なう場合はこのコールゲートを呼

```
GotoProtMode    endp
;** MACRO _GotoRealMode <dst>
;** desc : back to real mode and goto <dst>
;**        CS := _TEXT, DS, ES, SS := DGROUP
;**        SS:(E)SP points to same addr before and after mode switching
_GotoRealMode   macro   dst
                push    word ptr (offset dst)
                db      66h, 0eah           ;; use16 jmp far ptr GotoRealMode
                dw      offset GotoRealMode, SEG_64K_CODE
                endm
;** MACRO _CallRealMode <dst>
;** desc : call real mode routine at <_TEXT>:<dst>
;**        <dst> routine must be return with 16bit-near-ret instruction
_CallRealMode   macro   dst
                local   ProtReturnAddr
                push    word ptr SEG_GX_CODE ;; simulate 32-bit mode far call
                push    word ptr (offset ProtReturnAddr - offset TextGxOffset0)
                push    word ptr (offset GotoProtMode)   ;; return addr
                push    word ptr (offset dst)
                db      66h, 0eah           ;; use16 jmp far ptr GotoRealMode
                dw      offset GotoRealMode, SEG_64K_CODE
                align   4
ProtReturnAddr  label   near
                endm
;** actual mode switching code **
                align   4
GotoRealMode    proc    near
                pushf
                cli
                mov     ProtModeSegDS,ds ; save protect mode segment
                mov     ProtModeSegES,es
                mov     ProtModeSegSS,ss
                push    eax             ; save eax
                mov     ax,SEG_64K_DATA ; setup segment descriptor cache
                mov     ds,ax           ;   for MS-DOS real mode
                mov     es,ax
                mov     ss,ax
                db      066h            ; 32-bit form LIDT inst
                lidt    IdtRealPtr      ; load real mode IDT
                mov     eax,cr0         ; protection disable
                and     al,0feh         ; clear bit-0 (PE)
                mov     cr0,eax
                db      0eah                    ; jmp far ptr GotoRealMode1
                dw      offset GotoRealMode1, _TEXT ; flush queue & setup CS
                align   4
GotoRealMode1:  mov     ds,RealModeSegDS ; setup real mode segment
                mov     es,RealModeSegES
                mov     ss,RealModeSegSS
                pop     eax
                popf
                ret                     ; near return
GotoRealMode    endp
```

```
;** Main routine **
begin:          mov     ax,DGROUP
                mov     ds,ax
                mov     ss,ax
                lea     sp,StackTop
                cli
                call    ProtectSetup    ; setup misc data structure
                call    EnableA20       ; enable A20 line
                _GotoProtMode   ProtSampCode
GxTerminate:    sti                     ; GX end, call from CALL-GATE
                mov     dx,offset EndStr ; ending message
                mov     ah,09h
                mov     ax,4c00h        ; program terminate
                int     21h
;
KeyInput        proc    near            ; read char without echo back
                mov     ah, 08h
                int     21h             ; result char --> AL
                ret                     ; return to protect mode
KeyInput        endp

;== ProtectSetup : Setup misc data for protect mode management ==
ProtectSetup    proc    near
;** setup for LGDT, LIDT instruction
                xor     ebx,ebx         ; compute linear address of GDT base
                mov     bx,ds
                shl     ebx,4
                lea     eax,GdtBase[ebx]
                mov     [GdtPtrBase],eax
                lea     eax,IdtBase[ebx] ; linear address of IDT base
                mov     [IdtPtrBase],eax
;** setup each segment base
_CompLinearAddr macro   seg,off    ;; compute linear address, seg:off ===> eax
        ifb     <off>
                xor     eax,eax
            ifnb <seg>
                mov     ax,seg
                shl     eax,4
            endif
        else
                xor     eax,eax
                mov     ax,seg
                shl     eax,4
                lea     eax,off[eax]
        endif
                endm
_SetDescBase    macro   seg,off,dst        ;; setup each field
                _CompLinearAddr seg,off         ;; linear addr ==> eax
                mov     [dst]._BAS_0_15,ax
                shr     eax,16
                mov     [dst]._BAS_16_23,al
                mov     [dst]._BAS_24_31,ah
                endm
```

び出すわけであるが，現在は単にプログラムを終了させるためだけに使用している．なお，LDTの特権レベルは現在０にしてあり，何の制約も与えない．将来はレベル３へ移行させる予定である．

それではプログラムについて説明しておこう．

一番重要なルーチンとしてはプロテクトモード↔リアルモードの切り替えルーチンがある．

リアルモードからプロテクトモードに切り替えるにはGotoProtModeルーチンを使用する．まずLGDT/LIDT命令でGDTとIDTをセットアップする．次にCR0レジスタのビット０を１にしてプロテクトモードに移行する．このあとJMP $ 2命令で命令キューをフラッシュしたあと，LLDT命令でLDTもセットアップする（LDTはGDT中の１エントリでなければならない）．その後，プロテクトモードのセグメントセレクタをロードすれば移行が完了する．

逆にプロテクトモードからリアルモードに切り替えるにはGotoRealModeルーチンを使用する．まず一番最初にUSE16セグメントに制御を移行しなければならない．次に各セグメントセレクタにセグメント長64Kbytesのセグメントをセットする．そしてIDTにはベースアドレス０，リミット3FFhをロードし，CR0のビット０を０にしてリアルモードへ戻る．次にJMP farでCSもセットアップするが，これでCPUは完全にリアルモードへ戻ることになる．

このGotoRealModeと似たルーチンとして，CallRealModeも用意した．これはリアルモードの指定されたルーチンを実行したあと，またプロテクトモードに戻ってくるものである．

IDTに登録された割り込み処理ルーチンの内容であるが（RealIntHandler），プロテクトモードで割り込みが発生すると，まずCPUをリアルモードに切り替えてから同じ割り込みをINT xxh命令で再発行している．その後CPUをプロテクトモー

```
;** setup each descriptor in GDT **
                _SetDescBase    DGROUP,GdtBase,GdtDescGdt
                _SetDescBase    DGROUP,LdtBase,GdtDescLdt
                _SetDescBase    DGROUP,LdtBase,GdtDescLdtAL
                _SetDescBase    DGROUP,IdtBase,GdtDescIdt
                _SetDescBase    ,,GdtDescFlat
                _SetDescBase    _TEXT,,GdtDesc1MCode
                _SetDescBase    DGROUP,,GdtDesc1MData
                _SetDescBase    _TEXT_GX,,GdtDescGxCode
                _SetDescBase    DGROUP,,GdtDescGxData
                _SetDescBase    _TEXT_USER,,LdtDescUserCode
                _SetDescBase    _DATA_USER,,LdtDescUserData
                ret
ProtectSetup    endp
;
EnableA20       proc    near
        if _SYSTEM_PC9801
PORT_A20        =       0f2h
                out     PORT_A20,al
        endif
        if _SYSTEM_PCAT
                mov     al,0d1h         ; this is very buggy method, sorry
                out     064h,al
                mov     cx,-1
                loop    $
                mov     al,0ffh
                out     064h,al
                mov     cx,-1
                loop    $
        endif
                ret
EnableA20       endp

;** Real mode interface routine, call by 'ProtIntHandler' **
;    get intterrupt number, issue INT ??h, return to protect mode
RealIntHandler  proc    near
                push    ax
                mov     al,2+6+8[esp]   ; get interrupt vector number
                                        ; 2 := 'push ax'
                                        ; 6 := '_CallRealMode' return information
                                        ; 8 := pushed 2 segment registers
                mov     byte ptr cs:RIHIntVector,al
                jmp     short RealIntHandler1   ; flush instruction queue
                align   4
RealIntHandler1:
                pop     ax
                db      0cdh            ; INT ??h
RIHIntVector    db      ?
                ret                     ; near return
RealIntHandler  endp

_TEXT           ends
```

```
                assume  cs:_TEXT_GX
_TEXT_GX        segment
;** Protect mode Interrupt handler **
_ProtIntHandler macro   n
IdtEntry&n&:    push    dword ptr n             ;; push interrupt number
                jmp     near ptr ProtIntHandler ;; re-issue it in real mode
                endm
_IdtEntryNumber =       0
        rept    256                     ;; prepare for INT 00h -- INT 0FFh
                _ProtIntHandler %_IdtEntryNumber
_IdtEntryNumber =       _IdtEntryNumber + 1
        endm
ProtIntHandler  proc    near            ; protect mode interrupt handler
                push    ds
                push    es
                _CallRealMode   RealIntHandler  ; re-issue it in real mode
                pop     es
                pop     ds
                add     esp,4           ; discard interrupt number (dword)
                iretd                   ; 32-bit mode IRET
ProtIntHandler  endp

;** Protect mode System call handler **
SystemCall      proc    far
                _GotoRealMode   GxTerminate     ; terminate immediately
SystemCall      endp

;** Protect mode sample **
ProtSampCode:   sti
        mov     bl,10
loop1:          _CallRealMode KeyInput  ; read char without echo back
        mov     cl,10
                mov     dl,al
                mov     ah,02h
loop2:          int     21h             ; write char, direct call to MS-DOS
                inc     dl
        dec     cl                      ; inner loop done ?
        jne     loop2
                mov     dl,0dh          ; out CR/LF
                int     21h
                mov     dl,0ah
                int     21h
        dec     bl                      ; outer loop done ?
        jne     loop1
;
                _SystemCallGate         ; program end
;
_TEXT_GX        ends

                end     begin
```

ドに戻し，IRETによって割り込みの処理を終了する．ハードウェア割り込みなどはすべてリアルモード側に任せている．

プロテクトモードからたとえばINT 21hを実行すると，リアルモードのMS-DOSを呼び出すことができる．ただしセグメントレジスタはプロテクトモードとリアルモードではまったく解釈が異なるので，セグメントレジスタを引数とするようなMS-DOSコールは使用することができない（これを解決するのが本来のDOS-Extenderである）．

それではプログラムの流れを順に追っていこう（ラベルbeginから実行開始）．まず最初にリアルモードの各セグメントをセットしたあと，ProtectSetupルーチンでLGDTやLIDT命令用のオペランドや，GDT，LDT内の各セグメントディスクリプタをセットアップし，EnableA20ルーチンでアドレスラインA20のマスクを解除する．この後，プロテクトモードのProtSampCodeへジャンプして，プロテクトモードで実行を開始する．

ProtSampCodeでは，CallRealModeを使って，リアルモードのKeyInputルーチンを呼び出して，キーボードから1文字入力する．次に，プロテクトモードから直接INT 21h, AH＝02hでMS-DOSを呼び出し，文字を10回エコーバックする．セグメントレジスタを使わないファンクションコールだからできる芸当だ．

最後にコールゲート（SEG_SYSCALL）を呼び出して終了する．

なお，今回のサンプルでは，LDTをセットアップしているが，プログラム中では使っていない．

今後の予定

今回はサンプルがやや長くなってしまった．しかし内容は単純であり，プロテクトモードといっても別に難しくないということが分かっていただけたと思う．次回はこれを進化させ，より実用的なDOS-Extenderに仕上げる予定である．

Emacs Lisp入門

第1回 Lispによる作業環境の構築

半田 剣一，小方 一郎

UNIX環境の標準エディタとなりつつある，GNU Emacsの最大の特徴は，徹底的なカスタマイズが可能なことです．標準で用意されているEmacsの環境は，Emacs Lispでのプログラムによって実現できる環境の1つの例でしかありません．単なるエディタとしてだけでなく，コンピュータを使ううえで発生するさまざまな目的に合わせたインターフェイスとして，使うことができるのです．Emacs Lispによってこのような作業環境を構築する方法を，Emacs初心者に向けて説明します．

1 この連載の目的

この連載を読んでいらっしゃるみなさんは，もうGNU Emacsやそれを日本語化したNemacsをお使いのことと思います．

そこで，今回はGNU EmacsやNemacsは一通り使えるようになったけれど，この先カスタマイズ（自分の使い方に合わせた仕様の変更）はどうやってやっていくのだろう？ また，バッファやファイルの内容をプログラムで処理しようと思うのだけれど，どのようにすればいいのだろう？ といった人たちへのガイドをしようというわけです．

つまり，この連載はGNU Emacsの使いこなしがテーマです．GNU Emacsを使いこなすにはEmacs自体の記述言語であるEmacs Lispと呼ばれるLispの方言をマスターしなくてはなりません．そこでこの連載のタイトルがEmacs Lisp入門となりました．

また，NemacsはGNU Emacsを日本語化したものですから，拡張の仕方なども同じようにEmacs Lispを通してできるようになっています．つまり，日本語を取り扱えるようにEmacs Lispの関数やグローバル変数を拡張してあるのです．

この連載では，次のようなことをできるようになるのが目標です．

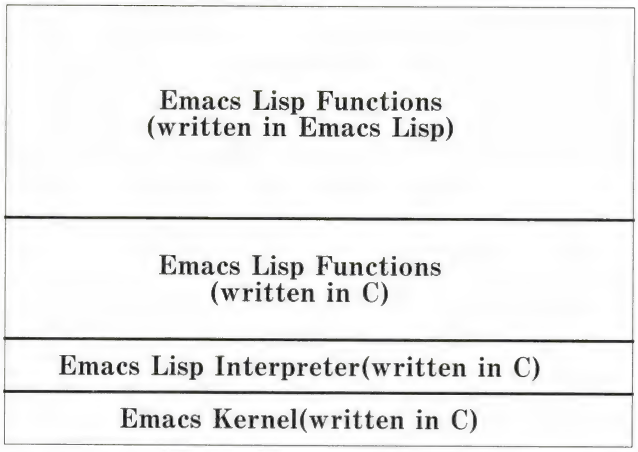
- ●インタープリタの使い方
  GNU Emacsの中でLispインタープリタが使えるようになる．
- ●カスタマイズの仕方
  GNU Emacsを自分の好みの仕様に変更できるようになる．
- ●キーマップ
  Emacsでよく使うコマンドを，簡単に起動できるように自分の好きなキーに割り付けられる．
- ●テキスト処理
  バッファ内でのテキストの挿入，削除，正規表現による文字列の探索ができる．
- ●バッファとウィンドウ
  バッファとウィンドウを使いこなす．バッファの例としては，rmail用のバッファRMAILが挙げられます．rmailは電子メールの取り扱いのためのコマンドです．みなさんもすでにお使いかもしれませんね．rmailはRMAILという名前のバッファを作り，その中でメールを扱うコマンドを使えるようにします．
- ●メジャーモードとマイナーモード
  自分でモードを設計し，作成できるようになる．マイナーモードの例としてはauto-fillモードがあります．このモードでは入力した文字数が規定の幅（fill-columnという変数に設定されている）に達すると自動的に折り返してくれるモードです．この原稿はLaTeXというメジャーモードとauto-fillというマイナーモードの

Copyright 1990 Kenichi Handa, Ichirou Ogata, Ascii Corp. 著作権表示とこのnoticeが含まれている限り，この記事「Emacs Lisp入門」をこのまま写して配布することを許可します．

もとで書いています.

●プロセスとの通信
UNIX の他のプロセス（ツール）との通信をするバッファを設計し，作成する．つまり，GNU Emacs の中から自分の設計したユーザーインターフェイスで UNIX ツールを使えるようにする．たとえば shell というコマンドは csh と通信することにより Emacs の中からシェルを使えるようにするものです．また，先ほどの rmail も Emacs 外部のプログラム movemail や sendmail と通信してメールのやり取りをするのです．

図1 Emacs プログラムの階層構造



GNU Emacs は，システムとのインターフェイスを担当する C の関数群，Emacs Lisp を実現するための C の関数，さらに Emacs 自身を実現する Lisp の関数群よりなっている．

しかし，これらの機能をただ説明していくだけでは面白くありませんので，全体を通して何か簡単な Emacs Lisp によるアプリケーションを作ることにしましょう．上記の機能をだいたいまんべんなく含んでいるアプリケーションとなると，やはり UNIX 上のツールのユーザーインターフェイスということになります．ここでは bib 関連のコマンド*1を Emacs の中から便利に使うためのツールを作ることにします．

連載の1回目である今回は，カスタマイズの仕方ぐらいまで話を進める予定です．

2 Emacs Lisp の使い方

■ Emacs Lisp とは何か

Lisp について何やらとても難しいもののように書いてあることもありますし，人工知能用言語とか，関数型言語とか難しげな効用を説く人もいますが，恐れることは何もありません．ちょっとばかり使いやすい，便利なプログラミング言語にすぎないともいえるのですから．

今回は，Lisp にまったく馴染みのない人でも理解できるように話を進めていきますから，安心してください．この連載を読んで Lisp へのアレルギー（もし，そういうものがあれば）を解消してもらえたらうれしい．

さて，「Lisp を勉強したくとも Lisp の処理系なんて持ってないよ！」と思われるかもしれません．でもそんな心配もいりません．実は Emacs は，Lisp インタープリタを備えているのです．というよりは，Emacs 自体が Lisp で動いているといったほうが正確です．

図1 に Emacs というプログラムの全体の構造を示しておきます．図の箱の面積は大まかにソースプログラムの大きさを示しています．Emacs のコードの大半は Lisp 関数の定義に占められていることが分かります．

ではその Lisp が動いているようすを順次みてみます．実際に自分で試しながら読み進むとよいでしょう．

■ Emacs Lisp の呼び出し方

まず* scratch *バッファを呼び出してください．GNU Emacs を引数なしで立ち上げると，自動的にこのバッファに入っています*2（**図2**）．モードラインの見方は分かりますね．Lisp Interaction というモードになっているでしょう．つまりこの *scratch* バッファでは Lisp がインタラクティブに実行できるのです．このバッファに

```
(+ 1 2)
```

を打ち込んで Line Feed (C-j) *3を入れてください．Carriage Return (C-m) ではだめです．結果の 3 がバッファ内に得られます（**コラム1**）．

```
(message "Hello World!")
```

も試してみてください．message は Mode Line の下の Echo Area と呼ばれる場所（要するにディスプレイの一番下です）に，引数として与えられた文字列を表示する関数です．

このほかのやり方として，普段，僕たちが簡単に Lisp インタープリタを呼び出す場合は，eval-expression というコマンドを使います．標準のキーバインドでは ESC-ESC で呼び出せるはずです*4．Eval: というプロンプトがミニバッファ内に出ますから，そこで，すかさず評価（実行）する式を打ち込みます．

＊1　bib (Bibliography) は UNIX 上の一連の文献データベース管理ツールを指します．しかしこれは文献に限らずいろいろなデータベースの管理に使えます．たとえばレコード（CD，LP）や住所録などのデータベースも扱えるのです．% man-k bib で調べると bib に関する一群のコマンドが見つかります．

＊2　引数なしで Emacs を立ち上げるとコピーライトその他のいろいろなメッセージが表示されますが，その内容が実際に* scratch *バッファに入っているわけではありません．何かキーを押せば（C-@でもよいです）メッセージは消えます．

＊3　Ctrl-j（コントロールキーと j キーを同時に押す）のことを，GNU Emacs ではこう書きます．ちなみに，M-x は Meta-x でメタキーと x を同時に押すことで，エスケープキーを押して離してから x キーを押すことで代用できます．

図2　GNU Emacsウィンドウの例

```
(+ 1 2)
3
(car '(1 2 3 4))
1
(cdr '(1 2 3 4))
(2 3 4)
(defun second (x) (car (cdr x)))
second
(second '(1 2 3 4))
2
█







[--]--**-NEmacs: *scratch*          8:22pm   (-JJ-:Lisp Interaction)--All-----
```

最初に作られるGNU Emacsのバッファ＊scratch＊では，Emacs Lispの関数を，簡単に評価（つまり実行）してみることができる．

■コラム1■

Lispの関数の呼び出し方

さて，

(+1 2)

というのがLispでの関数の呼び出し方なのです．+が関数名で1と2が引数です．Lispではあらゆる関数が必ず値を返します．

たとえば，C言語では

1+2＊3

なんていうふうに書きますけれど，Lispでは，

(+ 1 (＊ 2 3))

というふうに書くわけです．これは

(＊ 2 3)⟹6
(+1 6)⟹7

というふうに「評価」されます（関数が呼び出される）．はじめはちょっと面倒なように感じますが，慣れてください．

Eval: (message "I'm fine!")

このようにしてLispの式をEmacsの中で評価できるわけです．

■GNU EmacsはLispで動いている

ここで，実際にGNU EmacsはEmacs Lispで動いているというところをお目にかけましょう．まず，適当なテキストファイルをfind-file（C-x C-f）します．/etc/passwdとか，なんでもいいんです．ここでESC-ESCで呼び出したLispインタープリタで

(next-line 4)

を評価してみてください．カーソルが4行分下がるでしょう？（ちゃんと実行前のカーソルの位置を覚えていないとだめです）つまり，next-lineは「引数の行数だけ下に行く」関数なのです．

このほかに，

(goto-line 10)
(end-of-buffer)

なども実行してみて，結果を注意深く観察してください．GNU Emacsは編集に必要なこういうコマンドを山ほど持っているわけです．

■オンラインヘルプ

カスタム化可能，拡張可能といった性質のほかに，GNU Emacsはその中にドキュメントを調べる仕組みを備えています．みなさんもそのいくつかの機能はお使いかと思いますが，Emacs Lispでカスタマイズしたり本格的にプログラムしようという場合には，このドキュメントがとても役に立ちます．というより，このオンラインヘルプの機能がなければ不可能といってもいいでしょう．そこでGNU Emcasのヘルプ機能についておさらいしてみます．

前に（next-line）関数を実行してみましたね．この実行結果はC-nを4回押す，もしくはC-u C-nとやったのと同じだということに，気が付かれましたか？つまりC-nという「キー」はこの関数に「バインド」（結合）されていて，C-nを打つことで（next-line）が実行されます．

どのキーを打てばどの関数が起動されるのかを見るには，

1. describe key briefly (C-h c)
2. describe key (C-h k)

つまりC-h c C-nやC-h k C-nでいいのです．1.ですと簡単な説明

runs the command next-line

がEcho Areaに現われますし，2．だと＊Help＊バッファに詳しい説明が得られます．

また，その関数がどういう働きをするのかを知るには

- describe function (C-h f)

を使います．具体的にはC-h f next-line<CR>と入力するのです．next-lineの入力ではスペースキーによるcompletion機能も使えます[＊5]．似たような機能に

- describe variable (C-h v)

というのもあります．

さて，たとえば1行を消すにはどうすればよいのか，関数名も分からない，キ

＊4　標準の設定では，＊Help＊バッファが現われて，本当にこのコマンドを使いたいのか確認を求めてきます．ここでyキーを押すと，これ以降このコマンドが自由に使えるようになります．

＊5　completionは，読者の方ももう使っていらっしゃるかもしれませんね．これは，コマンド名／ファイル名などの入力時にその時点で可能な名前のリストをバッファ内に表示したり，もしくは，もうすでに決定している文字列を埋めてくれたりする機能です．関数の名前が全部は分からなくとも，可能性のある名前の中から選んでいけばいいわけです．これは，Emacsを使いこなすうえで大変便利な機能です．

ーも分からない，というときに便利なのが，アプロポス(apropos)機能です．アプロポスは，UNIX の ed エディタなどでも用いる正規表現とマッチする名前を探索する機能です．たとえば，行に関係する関数，変数を調べたければ，

M-x apropos＜CR＞ line＜CR＞

とやればいいわけです(line と英語で入れないといけないのが玉にキズですが)．ただし Emacs は行に関する多くの関数を持っていますから，これを実行すると時間がかかるかもしれません（図 3）．

3 簡単なカスタマイズ

ここまでくれば簡単なカスタマイズはできます．やり方は自分のホームディレクトリに.emacs というファイルを作り，その中に自由に Emacs Lisp プログラムを書くわけです．Emacs は立ち上り時に自動的にこのファイルをロードして実行します．

■ Emacs 変数の値の変更

たとえば僕は自分の.emacs の先頭に

```
(setq make-backup-files nil)
(setq load-path
  (cons
    (expand-file-name "~/elisp")
    load-path))
```

という行を置いています（コラム 2，3）．それぞれの変数，関数の意味は自分で describe-function や describe-variable を使って調べてください．

■キーの定義

キーを関数にバインドするには global-set-key という関数を使います．

```
(global-set-key "\er" 'rmail)
(global-set-key "\e=" 'what-line)
(global-set-key "\e-" 'goto-line)
```

などを実行してみてください．"\er" は ESC r を表わしています．最後の rmail 等の関数名に'（クォート）が必要です．クォートのところは，とりあえずこういうものだと思っていてください．

これによって，めでたく

ESC r : rmail
ESC= : what-line
ESC- : goto-line

と定義されたわけです．バッファ内の適当なところにカーソルを動かし ESC=と打てば，その位置がバッファ内の何行目

図 3　apropos の使用例

```
This is a test line.

The window below shows output of 'Apropos: line'.
█





[--]--**-NEmacs: *scratch*           3:01pm   (-JJ-:Lisp Interaction)--All-----
beginning-of-line            C-a
  Function: Move point to beginning of current line.
c-auto-newline
  Variable: *Non-nil means automatically newline before and after braces,
c-indent-line
  Function: Indent current line as C code.
center-line
  Function: Center the line point is on, within the width specified by 'fill-co\
lumn'.
colon-line-end
command-line
[--]-----NEmacs: *Help*           3:01pm   (-JJ-:Fundamental)--Top----------
```

画面の下半分に表示されている＊Help＊バッファに，apropos 機能で呼び出した，line に関する関数や変数のリストが説明付きで表示されている．

■ コラム 3 ■

リストと car，cdr，cons

Lisp ではリストというデータ型が使えます．リストは，（と）で囲まれた中に 1 番目から順番に要素を並べていきます．1，2，3 のリストは(1 2 3)と書きます．リストの先頭の要素を取る関数が car，先頭以外の要素を取る関数が cdr と呼ばれます．

(car '(1 2 3 4))⟹1
(cdr '(1 2 3 4))⟹(2 3 4)

ここで '(クォート)というのが出てきました．リストの形は関数の評価の形と同じものですよね．これを区別するためにクォートを付けるのです．たとえば，

(car (1 2 3 4))

だと 1 という名前の関数を 2，3，4 という引数で呼び出してしまうことになってしまいますから．

逆にリストを作り上げるには cons という関数を使います．

(cons 1 '(2 3 4))⟹(1 2 3 4)

となるわけです．

2 番目の要素を取る関数はどうなるでしょう？

(car (cdr '(1 2 3 4)))

でいいわけですね．

(cdr '(1 2 3 4))⟹(2 3 4)
(car '(2 3 4))⟹2

というふうに評価が進むわけです．もちろん，リストを要素とするリストも作れます．

(1 (2 3) (4 5))

これは 3 つの要素を含んだリストで，1 番目の要素が 1 で，2 番目の要素が（2 3），3 番目の要素が（4 5）となっているわけです．

■ コラム 2 ■

setq と変数

変数には setq という関数で値を set します．また，値を呼び出すには(クォートを付けずに）そのまま書けばいいのです．C 言語との対比で見てみましょう．

C 言語	Lisp
a=1;	(setq a 1)
b=2;	(setq b 2)
c=a+b;	(setq c (+ a b))

なのかが表示されます.

■フックの使い方

あるモードに入る時点で実行するべき関数を指定することもできます. そのような関数はフックと呼ばれています. たとえばtext-modeに入るときに必ずauto-fill機能をオンにしたければ,

```
(setq text-mode-hook
    '(turn-on-auto-fill))
```

と定義すればいいのです. これはturn-on-auto-fillという関数をtext-modeのフックとして定義したわけです.

■個人用の簡単な関数の定義

プログラム1に, 簡単な関数を定義してみました. C-h f (describe function)などを活用して読んでください. sendmail用のバッファの中でC-c C-f C-cと打てば, 自動的にメイルのヘッダ部にCC: フィールドという行が挿入され, カーソルがそこに移動します. **プログラム1**は同様な機能をReply-to: フィールドについて実現しようというわけです.

プログラムでやっているのは, mail-reply-toという関数を定義して, それをC-c C-f C-rというキーに割り付ける作業です.

この関数の定義の中身には難しい関数が次々出てきます. でも, 初心者にいきなりこんなのを読ませるのか, と怒らないでください. 僕自身もこれをいきなり書き下したわけではないのです. 実はこれはmail-ccという関数(先のCC: フィールドを付けるもの) の定義をソースからコピーしてきて, ちょこちょこっと変更しただけです*6. mail-ccがこれでうまく動いているのですから, mail-reply-toもこんなもんで大丈夫だろうと, 気楽に考えていてください.

プログラム1

```
(defun mail-reply-to ()
  "Move point to end of Reply-to: field.
Create the field if none."
  (interactive)
  (expand-abbrev)
  (or (mail-position-on-field "reply-to" t)
      (progn (mail-position-on-field "to")
     (insert "\nReply-to: "))))
(require 'sendmail)
(define-key mail-mode-map
  "\C-c\C-f\C-r" 'mail-reply-to)
```

最初の8行が関数の定義で, orを制御のために使っている点がポイント. このプログラムはGNU Emacs General Public License(次ページ参照)に従って利用できます.

■ コラム4 ■

関数の定義

関数の定義をするにはdefunという関数を使います. リストの2番目の要素をとる関数secondを定義して実行してみましょう.

(defun second(x)(car(cdr x)))
(second '(1 2 3 4))⟹2

xが引数です. 引数が2個のときはカッコの中に並べます. 2個の引数の平均をとる関数averageを定義してみましょう.

(defun average (x y)"Return average of two arguments." (/ (+ x y) 2))

関数にはこのように"(ダブルクォート)で囲まれたドキュメントを付けることができます. この関数を定義した後,

C-h f average

を試してください. このドキュメントが読めます.

同じようにdefvarという関数を用いて変数の定義をすることができます.

(defvar one 1 "one is bound to 1")

これはoneという変数を定義しその初期値に1を与えるものです. 最後の文字列がドキュメントに対応しています. これも,

C-h v one

を試してみてください.

関数を定義したらそれをキーにバインドします. ただし, 上で説明したglobal-set-keyではなく, define-keyを使っています. この時点では, この関数を使えばmail-modeでだけこのバインドを有効にできると理解しておいてくだされば結構です. この直前のrequireは, もしsendmailという機能がまだロードされていなければそれをロードしろといっているのです. これによって, mail-mode-mapという変数が初期化されます.

4 次回の予定

次回は, キーのバインドの一覧表であるキーマップの解説から始め, テキスト処理の基本, さらに本格的なEmacs Lispのプログラミングへと進んでいきます. 最初に述べましたbibインターフェイスプログラムbib.elの部品も次回から作り始めます.

---

*6 mail-ccは$Emacs/lisp/sendmail.elの中にあります. それを知るための手順は簡単で, まずmail-modeでdescribe-keyによりC-c C-f C-cはmail-toに割り当てられていることが分かります. そこで

```
% grep 'defun mail-to' $Emacs/lisp/*.el
```

でそれがsendmail.elにあることが分かるわけです. あとはこのファイルをEmacsのバッファに読み込みカット&ペーストです. この一連の手順をもっと簡単に行なうためにタグという機能がありますが, これについては次回詳しく触れましょう. 上の$EmacsはEmacsがインストールされているディレクトリの意味で, 普通は$Emacs=/usr/local/emacsです.
その下のディレクトリの主なものは:

lisp Emacs Lispプログラム(*.el)とそれをコンパイルしたファイル(*.elc)があります.
src Cで書かれたソースがあります.
etc Emacs Lispから呼ばれる外部プログラムがあります. rmailが呼ぶmovemailなどもここにあるのです.

Emacsとは

●歴史的経緯

Emacsの原型は，Richard M.Stallmanによって1976年に作成された．これは，TECOという，各種の編集機能を実行する手続きを記述できるインタープリタ型言語によるマクロを持ったエディタ上で実現されており，動作する環境もUNIXではなかった．その後，カーネギーメロン大学のJames GoslingによってUNIX版が書かれたが，このときにTECOの代わりにLispが使用された．以来，EmacsはUNIX上の標準的なエディタとして広く使われるようになり，さまざまな版が生じたが，最も一般的なものはStallmanによってUNIX用に新たに書かれたGNU Emacsである．GNUとは，Stallmanが主宰するFree Software Foundationのプロジェクトであり，多くの優れたソフトウェアがここで作られ，ソース付きで無償で配布されている．

●概要

GNU Emacsは主にUNIX上で動作するフルスクリーンエディタであり，最近ではviに代わるUNIX上の標準エディタとしての地位を占めつつある．編集機能は強力であり，マルチウィンドウ，マルチバッファ機能を用いた複数ファイルの同時編集や，強力なアンドゥ機能が特徴である．また，主要な機能はすべてLisp（Emacs Lisp）によって記述されているため，ユーザーがLispを用いて新たな関数を定義することで，ほとんどすべての機能を変更できる．メール，ニュースなどのためのパッケージも使われている．またEmacsのバッファの中で，シェルなどのプロセスを動作させることも可能である．このため，ユーザーはUNIX上のほとんどすべての処理をEmacs上で行なうことができる．強力なカスタマイズ機能を持っているため，Emacsは単なるエディタではなく，基本的なコンピュータ使用環境として利用されることも多い．

●日本語化

GNU Emacsを日本語対応とし，日本語を含むファイルの編集を行なえるように拡張したのがNemacsである．Nemacsの特徴としては，複数の漢字コードを自由に扱えることがある．Nemacsではファイルからの入力の漢字コードとファイルに書き出す際の漢字コードを別々に指定することができる．（編集部）

GNU EmacsとFree Software

Free Software Foundationは，ソフトウェアをすべての人が共有できるようになることを理想としている．そのため，GNUのソフトウェアはすべて無料とされており，自由にコピーして配布できる．またソースが添付されており，ユーザーが自由に変更して利用できる．また，変更したものを配布することもできる．またGNUのソフトウェアはFSFによっては，なんの保証もなされない．

しかし，FSFはGNUのソフトウェアに対して著作権を放棄しているわけではない．GNUのソフトウェアまたは，それに手を加えたものを受け取った人は，くれた人，もしくは売った人に対してソースを要求する権利と，受け取ったものを自由に配布する権利を持っている．ソフトウェアの自由な流通を保証するために，おおむねこのような条件を設定し，これを変更することは禁止している．

FSFが定めている条件は，GNU Emacs General Public Licenseに書かれており，GNUのソフトウェアとともに配布されている．詳しくはアスキーネットPCSのsalon.hackersに，このLICENSEと，関連するファイルをアップロードしておいたので，そちらを参照していただきたい．なお，共立出版の『GNU Emacsマニュアル』にはこれらのファイルの日本語訳が掲載されているが，こちらはバージョンが古く，現在のものとは若干異なる点がある．（編集部）

●入手先●

GNU Emacsを入手するには，持っている人からコピーしてもらうのが最も簡単な方法であるが，メディア代などの実費を支払えばFSFから配布サービスを受けることもできる．FSFの住所は以下の通りである．

Free Software Foundation
675 Massachusetts Avenue
Cambridge, MA 02139 U.S.A.

国内では，JUNETから配布を受けることもできるし，日経MIXのlisting2/GNU/emacs会議にもアップロードされている．

GNUのCopyright表示

```
;; Copyright (C) 1985, 1986, 1987 Free Software Foundation, Inc.

;; This file is part of GNU Emacs.

;; GNU Emacs is distributed in the hope that it will be useful,
;; but WITHOUT ANY WARRANTY.  No author or distributor
;; accepts responsibility to anyone for the consequences of using it
;; or for whether it serves any particular purpose or works at all,
;; unless he says so in writing.  Refer to the GNU Emacs General Public
;; License for full details.

;; Everyone is granted permission to copy, modify and redistribute
;; GNU Emacs, but only under the conditions described in the
;; GNU Emacs General Public License.  A copy of this license is
;; supposed to have been given to you along with GNU Emacs so you
;; can know your rights and responsibilities.  It should be in a
;; file named COPYING.  Among other things, the copyright notice
;; and this notice must be preserved on all copies.
```

このCopyright表示と許可告知はGNU Emacsのすべてのソースファイルに付加されている．GNU Emacs General Public Licenseの条件に従って，コピー，改変，再配布が自由であると宣言されている点が重要である．

ウィンドウシステムのプログラミングパラダイム

MacintoshからX Window，さらにSTARまで ウィンドウシステム上のプログラムの本質を衝く

中川幸男

The X Window Systemにおける最新のプログラム環境であるwidgetについて，プログラムの基本となるオブジェクト指向プログラミングの概念とともに解説する．

Xt Intrinsics

前回はThe X Window System（以下X）について，Xlibとwidgetを使った場合のプログラミングパラダイムの違いを見た．今回はwidgetが，Xt Intrinsics（以下，単にXtと呼ぶ）を使ってどのように作られているかを見てみよう．なお，この説明にはOOP（オブジェクト指向プログラミング）の考え方が必要となるので，OOPについても前回よりも少し詳しく説明するつもりである．

X Toolkit

X Toolkit（widget Xt）の第一の目的は，ウィンドウシステムのプログラミングを簡略化することである．つまり，前回見たようにXlibで直接プログラムを書くのはかなり複雑であるし，できるだけ再利用可能な「部品」（すなわちwidget）を用意しておけば，それだけ少ない労力でアプリケーションを作ることができるというわけだ．しかもOOPの考え方を提供することにより，新たに必要になった機能に対応するプログラムだけ書き加える「差分プログラミング」が可能となるため，さらにアプリケーションプログラマの負荷を減らすことができる．

そして，アプリケーションの部品（スクロールバーやコマンドボタン，メニューなどのwidget）を作ったり使ったりするためのルーチン群を提供しているのがXtである．したがって，Xで動くアプリケーションは，Xtと（あらかじめ提供されている）widgetを合わせて使うのが普通である（場合によってはXlibも使う必要があるが，今回は詳細は割愛する）．

表1

オブジェクト	（外界からの）メッセージを受け取り，それに応じた反応をするもの．とりあえず操作の対象となるデータ群と思っていればよい．
クラス	共通の性質を持つオブジェクトをひとまとめにしたもの．つまり，すべてのオブジェクトは必ずどれかのクラスに属している．ただし，クラスをどのように定義するかは非常に難しい問題である．
インスタンス	あるクラスに属するオブジェクトの実体．たとえばこの正方形□は正方形というクラスのインスタンスである．
メッセージ	オブジェクトに対して送られる指示／指令のようなもの．
メソッド	（メッセージに対する）オブジェクトの振る舞いを定める処理手続き．
インヘリタンス	生物の遺伝のように親の性質を子が継承すること．
スーパークラス	あるクラスに対する親のクラス．
サブクラス	あるクラスの子に相当するクラス．

Class and Instance

さて，widgetはOOPの考え方を実現しているわけだが，OOPに関する説明をするためには，どうしても普段聞き慣れない用語がたくさん登場するので，少しそのへんの話をしておこう（**表1**参照）．

OOPの考え方の基本は，前回も触れたように「オブジェクト」に対して「メッセージ」を送ると，それに対する「結果や反応」が得られるということだ．この

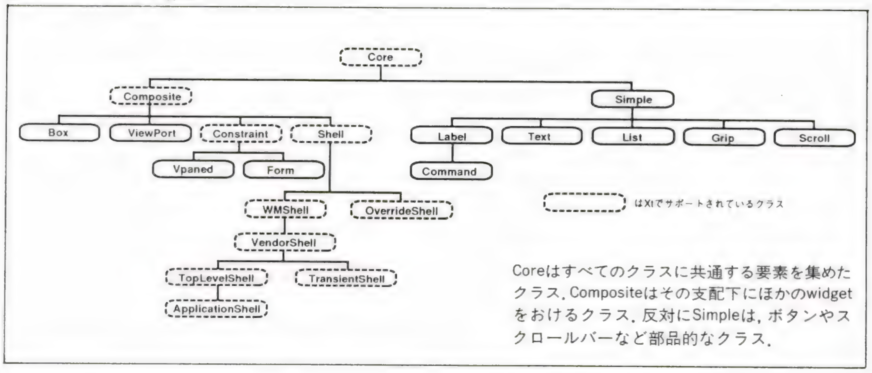
送られたメッセージをどう解釈するかを規定する手続きを「メソッド」と呼ぶ.

ところで,「オブジェクト」には共通の性質を持つものが多くある(たとえば,正方形も長方形も共に四角形という共通の性質を持っている).この共通項を「クラス」としてまとめることができる.逆にあるクラスの中のオブジェクトの実体を「インスタンス」と呼ぶ.

インスタンスは「インヘリタンス(継承)」によって,その属するクラスの持つ性質を継承することができる.また,クラスは階層的に定義することもでき,あるクラスの親に当たるものを「スーパークラス」と呼び,逆に子供に相当するものを「サブクラス」と呼ぶ.インスタンスと同様にサブクラスもそのスーパークラスの性質を継承することができる.このため,一般にクラスの階層が下がれば下がるほど機能は高くなるのが普通である.

ところで,あるクラスもまたオブジェクトである.したがって何らかのクラスに属しているわけだが,クラスが属するクラスのことを「メタクラス」と呼ぶ.メタとは1段階上の概念を表わす言葉である.よく「議論のための議論」という表現を使うが,OOPを知っている人は「メタ議論」などという言葉を使う.ただし,メタクラスのクラスは? などと考えていくと無限に続いてしまうので,通常は1段階だけにとどめている.

図1 Xawのクラス構成



Class of widget

ひととおり用語の使い方を理解していただいたので,具体的な話に移ろう.前回も少し触れたが,widget setはアプリケーションプログラマに対してOOP風の環境を提供してくれる.つまりwidget setはクラスの集合によって成り立っているのだ.現在までにXawやOPEN LOOK,Motifなどといったさまざまなwidget setがリリースされているが,それぞれ少しずつ異なったクラス構成を持っている.しかし,基本的な考え方はどれでも同じなので,今回はどのXのディストリビューションにも必ず含まれているXawを中心にお話ししよう.なお,Xtもwidgetを作るための基本的なwidgetクラスをいくつか定義している.

ちなみに,C言語自体は言語仕様としてはオブジェクト指向を特に意識していないために,クラスやメッセージの実現には工夫が必要である.筆者の知る限りではクラスは構造体(struct)として定義され,メソッドはその構造体の中に関数へのポインタとして宣言するのが普通である.

図1にXawのクラス構成を示す.一番基本となる部分はXtで規定されているものでCoreクラスと呼ばれ,以下のクラスすべてに共通する基本的な性質を規定している.具体的にCoreクラスがどのように定義されているかを見てみよう.ここではページの都合でCoreクラスの説明だけにとどめるが,以下の話はすべてのwidgetに共通することである.

まず,すべてのwidgetは「クラスパート」と「インスタンス指定パート」と呼ばれる2つの基本的な構成要素から成り立っている.このそれぞれがデータとメソッドのポインタ(これがメッセージに該当する)を含んだCの構造体として実現されている.また,この構造体のことをそれぞれ「クラスレコード」と「インスタンスレコード」と呼ぶ.

Class Record and Instance Record

クラスレコードは,そのクラスのすべてのwidgetに共通するデータとメソッドを含んでおり,このデータはそのクラスに属するすべてのwidgetに共有されるため,個々のwidgetの状態に直接関係するようなデータは含んではいけない(そのようなデータはインスタンスレコードに含めるべきものである).

```
typedef struct _WidgetClassRec  {
  CoreClassPart core_class;
}  WidgetClassRec, *WidgetClass;
```

これがCoreクラスのクラスレコードである.CoreClassPartは**リスト1**のように定義された構造体である.細かい説明は割愛するが,基本的にクラスデータとメソッドへのポインタが含まれたものと考えればよい.

インスタンスレコードはクラスレコードとは対照的に,個々のwidgetインスタンスがそれぞれ専用の実体を持つもので,現在のwidgetの状態を保持するための構造体である.以下にCore widgetのインスタンスレコードを示す.

```
typedef struct _WidgetRec  {
    CorePart core;
}    WidgetRec, *Widget;
```

同様に,CorePartは**リスト2**のように定義されている.上の2つの構造体を見ると分かるが,クラスレコードにはそのスーパークラスへのポインタ(superclass)が,そしてインスタンスレコードには自分自身へのポインタ(self)とクラスレコードへのポインタ(widget_class)が含まれている.

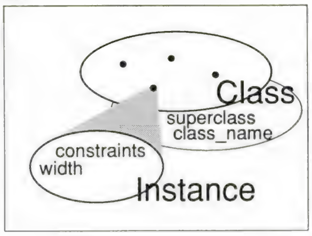
このようにして,それぞれのwidgetがどのクラスに含まれ,またクラスの階層のどこに位置するのかを表わしているわけだ(**図2**).

リスト1 クラスレコード

```
/* /usr/src/X11R4/mit/lib/Xt/CoreP.h より引用 */
typedef struct _CoreClassPart {
    WidgetClass       superclass;
                          /* pointer to superclass ClassRec */
    String            class_name;
                          /* widget resource class name */
    Cardinal          widget_size;
                          /* size in bytes of widget record */
    XtProc            class_initialize;
                          /* class initialization proc */
    XtWidgetClassProc class_part_initialize;
                          /* dynamic initialization */
    XtEnum            class_inited;
                          /* has class been initialized? */



    XtStringProc      display_accelerator;
                          /* display your accelerator */
    XtPointer         extension;
                          /* pointer to extension record */
} CoreClassPart;
```

傾向として，クラスレコードにはオブジェクト指向プログラミングを実現するための情報，インスタンスレコードにはwidgetがウィンドウを開いて活動するための情報が格納されている．

図2



リスト2 インスタンスレコード

```
/usr/src/X11R4/mit/lib/Xt/CoreP.h より引用 */
›edef struct _CorePart {
    Widget          self;
                        /* pointer to widget itself */
    WidgetClass     widget_class;
                        /* pointer to Widget's ClassRec */
    Widget          parent;
                        /* parent widget */
    XrmName         xrm_name;
                        /* widget resource name quarkified */
    Boolean         being_destroyed;
                        /* marked for destroy */
    XtCallbackList  destroy_callbacks;
                        /* who to call when widget destroyed */
    XtPointer       constraints;
                        /* constraint record */
    Position        x, y;
                        /* window position */
    Dimension       width, height;
                        /* window dimensions */
    Dimension       border_width;
                        /* window border width */
    Boolean         managed;
                        /* is widget geometry managed? */
    Boolean         sensitive;
                        /* is widget sensitive to user events */
    Boolean         ancestor_sensitive;
                        /* are all ancestors sensitive? */


                        /* window background pixmap or NULL */
      Boolean         visible;
                        /* is window mapped and not occluded? */
      Boolean         mapped_when_managed;
                        /* map window if it's managed? */
} CorePart;
```

Inheritance

さて，**図1**を見るとCoreクラスの下にSimpleクラスとCompositeクラスの２つがサブクラスとして定義されている．これらはインヘリタンスによってCoreクラスの性質をそっくり継承するわけだが，実際にはどのようにインヘリタンスが実現されているかを見てみよう．

ひとことでいえば，それぞれのサブクラスのクラスレコード内にそのスーパークラスのクラスレコードを，そっくりそのまま含んでしまえばよいということになる．試しにSimpleクラスのクラスレコードを見ると，以下のように定義されている．

```
(/usr/include/X11R4/mit/lib/Xaw/SimpleP.hより引用)
typedef struct _SimpleClassRec {
    CoreClassPart core_class;
    SimpleClassPart simple_class;
} SimpleClassRec;
```

それでSimpleClassPartでは，Coreクラスとの差分だけを以下のように定義している．

```
typedef struct {
    Boolean (*change_sensitive)
                    (/* widget */);
} SimpleClassPart;
```

つまり，Coreクラスですでに定義されている部分はそのまま利用して，新たに追加する部分だけを定義すればよいということだ．これがOOPの概念(特にインヘリタンス)を利用するうえで最大の利点といっても過言ではないだろう．

インスタンスレコードについてもまったく同様に継承を実現できる．

```
typedef struct _SimpleRec {
    CorePart core;
    SimplePart simple;
} SimpleRec;
typedef struct {
    /* resources */
    Cursor cursor;
    Pixmap insensitive_border;
    /* private state */
} SimplePart;
```

つまり，SimplePartの中ではSimpleクラスのインスタンスが使用するリソースの宣言のみを行ない，その他のインスタンスデータはそっくりそのままCoreクラスのものを流用できることになる．また，継承のもう１つの利点は，スーパークラスによって定められたオブジェクト(この場合ウィンドウと考えると分かりやすいだろう)の振る舞いをそのサブクラスすべてが受け継ぐという点だろう．つまり，個々のサブクラスが独自の振る舞いを規定するメソッドを定義しないかぎり，スーパークラスでの規定がそのまま適用されることになり，ユーザーインターフェイスの統一に効果を発揮することになる．

当然のことながら，すでにあるクラス階層の中に新たなwidgetクラスを追加することも可能である．実際にXawなどは，Xtが提供しているごく基本的なwidgetクラスにいくつかのクラスを追加する形で作られている．このときにも継承は非常に効果を発揮する．一般にオブジェクト指向ではないごく普通のプログラミング環境ですでに存在するプログラムに新たに機能を追加するのは容易なことではない．いかにモジュール分割を考え，再利用可能なライブラリ群を作ったとしても，現在あるものとほとんど機能は同じ

だが，少しだけ異なる仕様のサブルーチンなどを作ろうとすると，結局はコピーを作り，そのコピーに修正を加え，新たにライブラリに登録するといったような作業になってしまう．

ところがOOPの考え方を使えば，ごく自然にこうしたことが実現できる．しかもプログラミングのオーバーヘッドは極力抑えることができる．これはやはりウィンドウシステムに限らず多くのアプリケーションに有効な，非常に魅力的なプログラミングパラダイムだと思う．

Callback and Event Handler

余談はこのくらいにして，widgetとアプリケーションの関係について少し考えてみよう．まず，ユーザーインターフェイスをつかさどる部分(すなわちwidget)とアプリケーションの機能を実装する部分は，それぞれ独立していたほうが機能拡張などにも便利である．しかし，どこかでこの2つが結び付かないことには最終的に動くものにはならない．Xtではこのために大きく分けて次の2つの手法が用意されている．

1つは前回も少し説明したコールバック関数を使う方法である．つまり，あらかじめwidgetによって定義された状況が発生したときに行なうべき処理をアプリケーションが定義してwidgetがそれを呼び返す(だからcallbackと呼ぶ)という方法だ．そのためにはアプリケーションからwidgetに対してコールバックしてもらう関数を登録しておく必要がある．Xtの場合には以下の関数を使えばよい．

```
XtAddCallback(widget,callback_name,
                    procedure, client_data)
```

また，widgetの中にはコールバック関数を保存するためのコールバックリストと呼ばれる領域が確保されている．上の関数の2番目の引数にその名前を指定する．たとえば上のCoreクラスにはdestroy callbacksというXtCallbackList型のメンバがあるのがそれである．つまりこの関数はどのwidgetの何というコールバックリストにprocedureで指定されたコールバック関数を登録するかを指定することになる．最後のclient_dataというパラメータは，コールバック関数が呼ばれるときに引数として渡される．さらにコールバック関数はあらかじめ決められた形式でなくてはならず，以下のように定義されている．

```
void CallbackProcedure(widget,
                    client_data, call_data)
```

また，手法としてはコールバックとよく似ているがXで定められている特定のイベントに対して実行されるものを「イベントハンドラ」と呼び区別して扱っている．実態は上のコールバックと大差ないが関数の形式が違う．イベントハンドラの登録には以下の関数を用いる．

```
XtAddEventHandler(widget, eventmask,
        nonmaskable, handler, client_data)
```

また，イベントハンドラ自身は以下の形式で定義しなくてはならない．

```
void handler(widget, client_data, event)
```

Xでいうところのコールバックとイベントハンドラの大きな違いは，前者があらかじめwidgetが用意してくれたコールバックリストに関数を登録するのに対し，後者はwidgetの製作者の意図に関係なく，マウスやキーボードなどからのイベントをダイレクトに指定して処理するという点だろう．

Event Translation

もう1つの手段はアクションと呼ばれる．これは，コールバックの概念をより発展させたもので，ユーザーの「アクション」から次に実行すべき処理を「トランスレーションマネージャ」との協調によって決定するメカニズムである．こう書くと何のことだか分かりにくいかもしれないが，要は「ユーザーが何らかの操作を行なったときに，どのように反応するかを決定するメカニズムの違いによって，名前が分かれている」と考えればよいだろう．つまりコールバックの場合にはあらかじめwidgetによって決められた事象が起きたときに，どうするかを指定したが，アクションではwidgetで提供されていなくても，自分で設定した事象が起きたときに動けるという点が異なる．

そのためにアクションテーブルとトランスレーションテーブルというものを使用する．アクションテーブルは次のように定義されている．

```
typedef struct _XtActionsRec  {
    char *string;
    XtActionProc proc;
}   XtActionsRec;
```

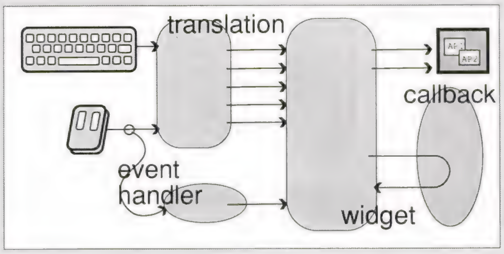
つまり関数名を表わす文字列と，その関数のアドレスを入れておく構造体である．このように関数名を文字列で登録しておくことにより，次に説明するトランスレーションテーブルを単純な文字列のテーブルにすることができる．そして次の関数を使って，このテーブルをトランスレーションマネージャに登録する．

```
XtAddActions(actions, num_actions)
```

トランスレーションマネージャの役割は，ユーザーからのアクションとwidgetまたはアプリケーションが用意したアクション関数とのマッピングをすることである．察しの良い読者はもうお分かりだと思うが，このマッピングを記述したものがトランスレーションテーブルである．つまりトランスレーションマネージャはトランスレーションテーブルを元にユーザーからのアクションを実際のアクション関数にマッピングしているわけだ．それによってたとえばマウスの真ん中のボタン(注：Xではマウスにボタンが3つあることを想定している)を押したら，アプリケーションを終了するといった関係を持たせることができる．さらに，トランスレーションテーブルはアプリケーション中にハードコーディングする必要はなく(しても構わないが)，.Xdefaults やアプリケーションデフォルトファイル(後述)に記述しておくことが可能である．したがってユーザーの好みに合わせてカス

図3

マウスのクリックなどサーバ側から上がってきた生のイベントは，event handler があれば，その処理を経たあとaction tableによって処理するwidget内部の関数が決定され呼び出される．widgetは，必要に応じてcallback procedure をサポートしている．



リスト3　event handler

```
/* sample event handler */

/* ARGSUSED */
static void
QuitEventHandler(w, client_data, event)
Widget  w;
caddr_t client_data;
XEvent  *event;
{
        exit(0);
}

XtAddEventHandler(widget_name, ButtonPressMask,
                  FALSE, QuitEventHandler, NULL);
```

リスト4　callback procedure

```
/* sample callback procedure */

/* ARGSUSED */
static void
QuitCallback(w, client_data, call_data)
Widget  w;
caddr_t client_data, call_data;
{
        exit(0);
}

XtAddCallback(widget_name, XtNcallback, QuitCallback, NULL);
```

リスト5　action table

```
/* sample action & translation */

/* ARGSUSED */
static void
QuitOnAction(w, event, params, num_params)
Widget  w;
XEvent  *event;
String  *params;
int     num_params;
{
        exit(0);
}

static XtActionsRec actionsTable[] = {
        {"quitOnAction", QuitOnAction}
};

XtAddActions(actionsTable, 1);
```

タマイズすることも，たやすくできるのだ．基本的なトランスレーションテーブルの書き方の例を以下に示す．

```
xhw*Label*translations:
<Key>Q: quitOnAction()
```

このように「実行ファイル名 widgetクラス名 translations: ユーザーアクション: アクション関数名」という形式で記述する．もし，特定の実行ファイルではなく，あるクラスに属するすべてのアプリケーションで共通の処理を指定したければ，最初の実行ファイル名は省略すればよい．要は後述するリソースとまったく同じ形式だと覚えておけばよい（つまりトランスレーションテーブルもリソースの1つなのだ）．

参考までに，前回紹介したxhw.cに，それぞれイベントハンドラ，コールバック，およびアクションの3つの方法でアプリケーションを終了させるためのルーチンを追加した例を載せておく（**リスト3，4，5，図3**）．なおLabel widgetはこのためのコールバックを持たないので，代わりにCommand widgetを使用する．

Resources

Macにもリソースという概念があったが，Xにも同様の考え方がある．呼び方も同じ「リソース」だが実体はかなり異なったイメージになっている．しかし，その目的はやはりカスタマイズにある．Xで扱うリソースにもMacの場合と同様さまざまな種類があるが，代表的なものとしてはウィンドウのID，カラーやフォント，テキスト情報やイメージ，さらにはウィンドウの位置やサイズといった情報が挙げられる．

これらのリソースは，いずれもユーザーが実行時にカスタマイズ可能なようにしておくべき情報である．また，Xではリソースの管理はリソースマネージャの仕事であり，XlibおよびXtのそれぞれにリソースマネージャの機能を利用するためのインターフェイスが用意されている．しかし，ユーザーが何も指定しなくても動けるように，アプリケーションは適切なデフォルト値を設定しておかなければならない．

それではどのようにしてリソースに初期値（すなわちデフォルト値）を与えるかを考えてみよう．まず一番最初に考えつくのはアプリケーション（もしくはwidget）内にハードコーディングしてしまう方法だ．これだと必ず初期値が定まるのでアプリケーションが動けなくなってしまうことはなくなるが，サイト単位，ユーザー単位のカスタマイズはできにくくなる．

そこで，Xtはもっと便利な手段を提供している．これが「アプリケーションデフォルトファイル」と呼ばれるもので，UNIXシステムでは通常/usr/lib/X11/app-defaults/ <class>という形で使われる慣例になっている．また，デフォルトファイルの名称もXtを初期化するためのルーチンXtInitializeに渡すクラス名と同一で最初の1文字（ただし最初がXで始まるクラス名の場合には2文字目まで）を大文字にした名前にすることになっている．さらに，このクラス名はアプリケーションの名前と同じにするのが普通だ．

たとえば前回紹介したxhwというプログラムの場合には，デフォルトファイル名はXHwとなる．また，xtermというアプリケーションの場合には，そのデフォルトファイル名はXTermという具合だ．そして，デフォルトファイルの中身は次のようになる．

xhw.label: How are you?

もしくは単に

*label: I'm fine, thank you.

詳細な説明は省くが，こうしてあると前回のxhwというアプリケーションをそのまま実行しても違う結果が得られることになる．また，この方法だと1台のマシンを複数のユーザーが使用する場合に，皆同じデフォルト値を使うことになる．そのため，各自のホームディレクトリの下に.Xdefaultsというファイルを持ち，そこに同様の書式で書いておくこともできる．さらに，アプリケーションの起動時にパラメータとして渡す方法がある．こうすれば，特定の場合だけ違ったリソースを設定することができる．いずれの方法を使うにせよ，ユーザーによるカスタマイズがリソースという概念によって可能になっているということを覚えておいていただきたい．

Programming with Xt

ここまでの話を整理する意味で，X Toolkitを使ってアプリケーションを作る場合の枠組みをもう一度見ておこう．どのようなアプリケーションを作る場合でも最低限以下の基本的なステップを実行しなくてはならない．

▷1 イントリンジックス(Xt)の初期化
▷2 widget(インスタンスの)生成
▷3 コールバックとイベントハンドラの登録
▷4 すべてのwidgetをリアライズする
▷5 イベントループに入る

なお，これはmain()の中身のみである．実際には必要なヘッダファイルをincludeしたり，コールバック関数などの実装も必要になるが，アプリケーションの本体であるmain()だけ抽出した．それ以外の部分はアプリケーションごとに異なるのが普通だからでもある．

最後にForm widgetとScroll widgetを使ったアプリケーションの例を示そう．**リスト6**は，前回紹介したxhw.cに若干の修正を加えてForm widget，Scroll widget，Command widget，そしてLabel widgetを使うようにしたものだ．実行結果を**図4**に示す．なお，このように，いくつかのwidgetを組み合わせて使う場合には，親となる widgetはCompositeクラスのものでなくてはならない．

まず，ヘッダファイルのincludeだが，Xtの機能を使うためにはIntrinsic.hが必ず必要である．次のStringDefs.hは，各種のリソース名が定義されており，コールバックなどのリソースを使う場合には必要となる．

その後のいくつかはいずれも使用しているwidgetのクラス定義を行なっているファイルで，それぞれのクラスに応じてincludeしなくてはならない．

次にあるのは各リソースの初期値の宣言である．これは，アプリケーションデフォルトファイルなどがまったくなかった場合に参照されるものである．もし，ほかにリソースを指定するものがあれば，そちらが優先される．

その次にいくつかの関数があるが，これらは基本的にコールバック関数の定義である．たとえば，scroll()という関数はScroll widgetで定義されているscrollProcというリソースに設定されるもので，スクロールバーの中でマウスボタンが押されたときにコールバックされる関数だ．中ではXtSetArg()とXtSetValue()というIntrinsicsで用意された関数を用いてlabel widgetに文字列を設定し，XawScrollbarSetThumb()という関数によってスクロールバーの中のサム(thumb)の位置や大きさを設定しているだけである．

さて，最後にmain()の説明をしよう．先に示したスケルトンと見比べてほしい．まず，▷1のXtの初期化をしなくてはならない．これをやっているのが，XtAppInitialize()という関数だ．これは今回説明しなかったが，アプリケーションコンテキストを使う場合の関数で，先に説明したfallback_resourceを使うためにこちらを使っている．通常はXtInitialize()を使えばよい．これにより，トップレベルのwidget(すなわち親となるウィンドウ)が作られる．

▷2はwidgetを作ることだ．そのためにはXtCreateManagedWidget()という関数を使う．この関数は引数で指定したクラスのwidgetのインスタンスを生成してくれる．また，第1引数で文字列を渡しているが，これは生成したwidgetの名前となるもので，リソースなどの設定のときに参照される．通常はそのwidgetのクラス名と同じ名前にするが，ここではあえて違う名前をつけているものもある．

そして，▷3のコールバックやアクション関数をそれぞれ該当するwidgetに登録する．そのためにはXtAddCallback()を使うのはここまでに説明したとおりだ．なお，その前にscroll()を呼んでいるが，これはlabel widgetに表示する文字列を初期化しているにすぎないので，あまり気にしなくてもよい．

▷4だが，widgetは生成しただけではインスタンスデータなどの初期化が行なわれるだけで実際に画面上には表示されない．そこで，XtRealizeWidget()を使ってトップレベルのwidgetをリアライズする．こうすると，その子のwidgetも同時に実体が作られる仕組みになっている．

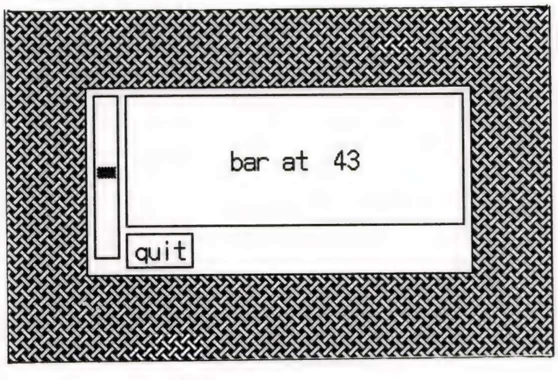
最後の▷5はおきまりのイベントループでおしまいである．ここではアプリケーションコンテキストを使っているためにXtAppMainLoop()を使っているが，通常はXtMainLoop()でも構わない．なお，この関数自体は無限ループとなっている．そこで，quitというコマンドwidgetをオマケで付けてみた．

以上でサンプルの説明を終わるが，widgetを使ったプログラミングの感覚をつかんでいただけただろうか？　文章で書くと結構複雑に見えるが，実際に自分でいくつかプログラムを書いて動かしてみると感じがつかみやすいと思う．もし，手近にXを使える環境がある方は，是非チャレンジしてみることをおすすめする．

リスト 6

```
#include <stdio.h>
#include <X11/Intrinsic.h>        /* standard Toolkit Header file. */
#include <X11/StringDefs.h>
#include <X11/Label.h>            /* Label widget */
#include <X11/Form.h>             /* Form widget */
#include <X11/Scrollbar.h>        /* Scrollbar widget */
#include <X11/Command.h>          /* Commadn widget */
#include <X11/Cardinals.h>

String fallback_resources[ ] = {
    "*form.X:              0",
    "*form.Y:              0",
    "*form.width:          256",
    "*form.height:         128",
    "*sbar.orientation:    vertical",
    "*sbar.length:         100",
    "*label.fromHoriz:     sbar",
    "*label.Width:         64",
    "*label.Height:        32",
    "*quit.fromHoriz:      sbar",
    "*quit.fromVert:       label",
    NULL
};

static int sbar_pos = 0; /* position of thumb */
static int sbar_len;     /* length of scroll bar */

/* call as XtNcallback from command widget */
static void quitProc(w, client_data, call_data)
Widget w;
caddr_t client_data, call_data;
{
    exit(0);
}

/* convert thumb position to Dimension */
static void getlen(sbar)
Widget sbar;
{
    Arg args;
    Dimension len;
    XtSetArg(args, XtNheight, &len);
    XtGetValues(sbar, &args, 1);
    sbar_len = (int) len;
}

/* call as XtNscrollProc from scrollbar widget */
static void scroll(sbar, label, posc)
Widget sbar;
Widget label; /* client_data */
int posc;
{
    Arg args;
    char buf[256];
    getlen(sbar);
    sbar_pos += (posc > 0)? 5: -5;
    if (sbar_pos < 0 ) sbar_pos = 0;
    if (sbar_pos > sbar_len ) sbar_pos = sbar_len;
    sprintf(buf, "bar␣at␣%3d", sbar_pos);
    XtSetArg(args, XtNlabel, buf);
    XtSetValues(label, &args, 1);
    XawScrollbarSetThumb(sbar, (float)sbar_pos / (float)sbar_len,
                         0.05);       /* thumb length = 5% */
}

/* call as XtNjumpProc from scrollbar widget */
static void jump(sbar, label, pos)
Widget sbar;
Widget label; /* client_data */
float *pos;
{
    Arg args;
    char buf[256];
    getlen(sbar);
    sbar_pos = (int) ((float) sbar_len * *pos);
    sprintf(buf, "bar␣at␣%3d", sbar_pos);
    XtSetArg(args, XtNlabel, buf);
    XtSetValues(label, &args, 1);
}

void main(argc, argv)
int argc;
char **argv;
{
    XtAppContext app_con;
    Widget toplevel, form, label, sbar, quit;

    /* open communication etc. */                                   ◁ STEP 1
    toplevel = XtAppInitialize(&app_con, "Sample", NULL, ZERO,
                    &argc, argv, fallback_resources, NULL, ZERO);

    /* create widgets */                                            ◁ STEP 2
    form = XtCreateManagedWidget("form", formWidgetClass,
                    toplevel, NULL, ZERO);
    sbar = XtCreateManagedWidget("sbar", scrollbarWidgetClass,
                    form, NULL, ZERO);
    label = XtCreateManagedWidget("label", labelWidgetClass,
                    form, NULL, ZERO);
    quit = XtCreateManagedWidget("quit", commandWidgetClass,
                    form, NULL, ZERO);

    /* initialize label */
    scroll(sbar, label, 0);

    /* add callback */                                              ◁ STEP 3
    XtAddCallback(sbar, XtNscrollProc, scroll, label);
    XtAddCallback(sbar, XtNjumpProc, jump, label);
    XtAddCallback(quit, XtNcallback, quitProc, toplevel);

    /* realize widget */                                            ◁ STEP 4
    XtRealizeWidget(toplevel);

    /* process events */                                            ◁ STEP 5
    XtAppMainLoop(app_con);
}
```

図 4



リスト 6 のプログラムの動作は，左側のスクロールバーを動かすと，右上の数値が変わる．スクロールするthumbは，マウス左右ボタンのクリックで５％上下し，中央ボタンによるドラッグで自由に移動できる．

Next Issue

どうも今回はwidgetを構成する基本的な要素の話だけでページが埋まってしまったようだ．次回はいよいよ自分でwidgetを作るというお話に突入する．そして，余裕があればオブジェクト指向プログラミングについてもう少しつっこんだ話もしてみたい．お楽しみに．

EP news

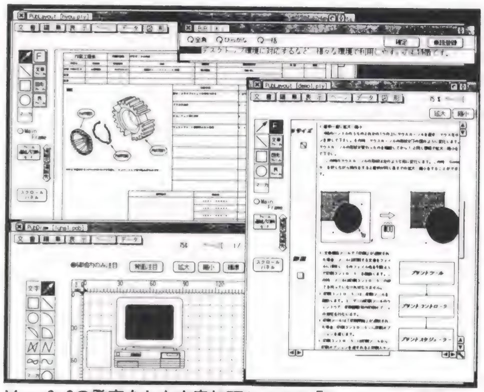
Electronic Publishing

基礎講座

文書交換のための
共通文書フォーマットの動向
―その3―「MDI」

キーワード

フォント表現フォーマット
IKフォーマット
INSTACフォーマット



Ver.3.0の発表をした文書処理システム「Publiss」の画面ハードコピー

基礎講座

文書交換のための共通文書フォーマットの動向　その3

文書の共有化，効率的文書交換を目指して

本講座では，「文書ファイルフォーマットの共通化」という課題に焦点を当てて，その具体的事例を紹介している．前号までに，ソニー㈱による「CDFF」，富士ゼロックス㈱による「MDI」を取り上げた．今号では，日本DEC㈱による「CDA」を紹介する．次回はこれらの総まとめとして，筆者がODAの最新動向とその位置付けに言及する．　（村上裕一）

CDA (Compound Document Architecture)

日本DEC㈱　企画本部
システムインテグレーテッドグループ

CDAの概要 開発の背景とゴール

コンピュータのビジネスへの浸透によって情報の質が大幅に変化し，また量も増大した．そして，その情報を表現する文書も，単なる文字情報の出力形態であったものが，コンピュータに蓄積されるさまざまなデータ――文字，図形，イメージ（写真，地図，イラストレーションなど），さらには，音声やフルモーションビデオなど――の出力形態へと変化してきている．つまり，文書は，もはや単なる“文を書き表わしたもの”ではなく，さまざまなデータを表現する“複合文書”(Compound Document)へと変化してきた．

さらに，コンピュータ産業界の飛躍的な発展の結果，コンピュータのユーザーは，上はメインフレームから，下はパソコンそしてワープロまでのさまざまな大きさの，しかも複数のベンダーのコンピュータを使用するに至っている．その複数のベンダーの多種多様なコンピュータに蓄積された多種多様な情報を1つの“複合文書”形態に統合したいというニーズに応えるという目的でCDAは，設計され開発された．

DECでは，1983年に，異なるデータを処理するアプリケーション間におけるデータ交換フォーマットの開発を，SARA (System Architecture and Review Authority――システムアーキテクチャ開発に関する超部門委員会）において発足し，DDIS (Digital Data Interchange Syntax――デジタルデータ交換シンタックス）を定義した．このDDISのメタシンタックスは，ISO (International Standard Organization――国際標準機構）のAbstract Syntax Notationのための標準であるASN.1 (ISO 8824とISO 8825の総称）をベースにして開発され，定義された．そしてCDAは，このDDISをベースに開発されたものであるから，国際業界標準であるISOのASN.1に準拠している．

◇CDA開発のゴール

1．ユーザーに，文字，テーブル情報（スプレッドシートや，データベースなどの），図形，イメージなどの文書に統合される情報をスムーズに処理する機能を提供すること．

2．将来における新しいデータタイプ（たとえば，音声，フルモーションビデオなど）に対応できること．

3．データ処理のアプリケーション間をリンクする機能を持つこと．

4．データや文書をユーザーアプリケーションのレベルで処理できること．

5．文書の処理を整然と定義されたレイヤ構造で提供できること．

6．できるだけ数多くの国際業界標準に準拠すること．

7．マルチベンダー環境において複合文書の処理ができること．

◇CDAの構成

CDAは次の構成要素から成り立つ．

1．データ交換フォーマット

2．システムサービス群とそれらに関連するAPI (Application Program Interfaces）群

3．CDAをベースにしたアプリケーション

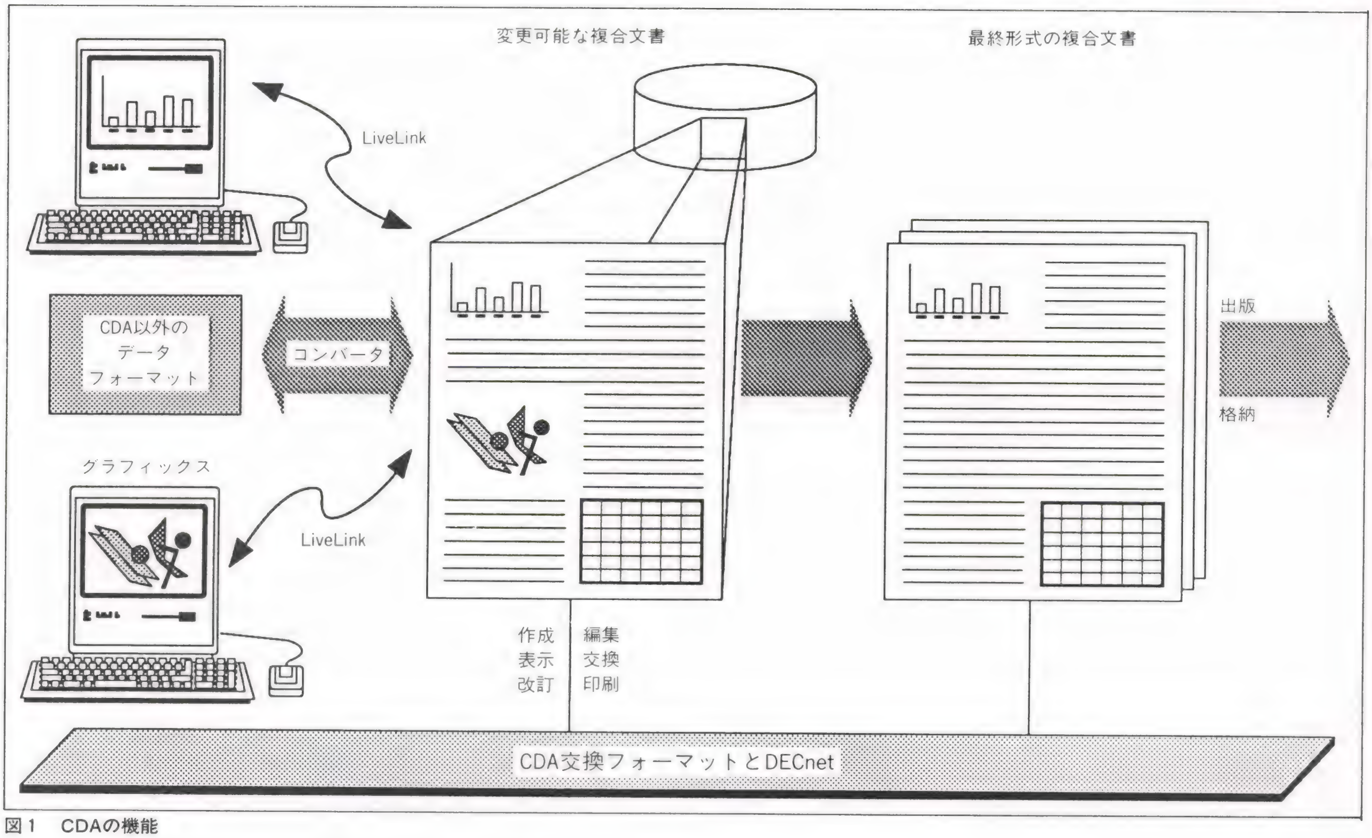


図1　CDAの機能

データ交換フォーマット

現在，２つのコード化標準（Encoding Standard；下記a，b）と３つの交換フォーマット（下記c，d，e）がCDAで使用されている．

a）DDIS（Digital Data Interchange Syntax――デジタルデータ交換シンタックス）

上述のISOのASN．1に準拠したDECの定義するデータ交換フォーマットのための基本となるコード化標準．

b）DOTS（Data Object Transport Syntax――データオブジェクト転送シンタックス）

DDISに準拠した複数の交換フォーマット内での，複数のデータ要素とその相互関連性のコード化のためのシンタックスで，電子メールプログラムにより複数の文書をその相互間のリンクを保持しながら交換する際に用いられる．

c）DDIF（Digital Document Interchange Format――デジタル文書交換フォーマット）

“変更可能な複合文書”（囲み参照）のCDA交換フォーマットである．このフォーマットは，CDAのツールキットによってサポートされ（後述），数種の製品によって処理され，DECの複合文書エディタであるDECwriteによって編集される．

d）DTIF（Digital Table Interchange Format――デジタルテーブル交換フォーマット）

変更可能なデータテーブルおよびスプレッドシートのCDA交換フォーマットであり，CDAのToolkitによってサポートされ，DECの製品であるDECdecisionの機能によって編集される．

e）DDFF（Digital Document Final Form――デジタル文書最終形式）

“最終形式の複合文書”（囲み参照）のCDA交換フォーマットであるが，現在のところ定義付けされていない．

DECでは，ISOにおいてSPDL（Standard Page Description Language――標準ページ記述言語）にCDAの最終形式に対する要求を提案し，ISOがその要求を極力取り入れて，SPDLが定義されるように積極的に活動している．ISOのSPDLの仕様が提供されるまで，DECにおいては，デファクトの標準ともいえるPostScriptを複合文書の最終形式として採用している．したがって，PostScriptの形式は，文書のフォーマットアプリケーションや，CDAToolkit内のデータ変換機能によって作成される．

DDIF

DDIFは，変更可能な形式で複合文書を作成，格納，配布するためのフォーマットである（この形式に対して，最終形式という形式がある――囲み参照）．DDIFでは文書を文書セグメントに順序付けした階層構造として表現する．

文書セグメント（あるいは単にセグメント）は，いくつかのコンテント，セグ

メント，セグメント属性の集まりとして定義する．コンテントには，テキストコンテント，グラフィックスコンテント，イメージコンテントなどがあり，そこに入れるデータの修飾情報としてセグメント属性がある．

表1にDDIFで使われる構成要素(CDAでは，Aggregate (集合体) と呼ぶ) の一覧を，また**図1**および**図2**に階層構造とソースリストを示す．

DTIF

DTIFは，変更可能なデータテーブル文書の作成，格納，交換に使われるフォーマットである．

DTIF文書に記述されるデータテーブルは，属性(Attribute)，列(Column)，行(Row)，セル (Cell)，ウィンドウ(Window)，領域 (Range) などで記述されている．

表2にDTIFの構成要素として，集合体の一覧を，**図3**，**4**に階層構造およびソースリストを示す．

APIをもとにしたサービス群

CDAのサービス群とそれに関連するAPI (Application Program Interface) は，アプリケーション内にCDAの機能を付加するために，アプリケーションの開発者によって使用される．

CDAサービス群に関連する標準のAPIは，DECの提供するすべてのシステム上に搭載されている．したがって，システムに対する依存性を取り除き，アプリケーションの移植性を保証している．

CDAツールキット

CDAツールキットは，CDAコンバータアーキテクチャの中核となるもので，基本OSやその他のアプリケーションに組み込まれているコンバータ群，あるいは，CDAコンバータライブラリとして提供されるコンバータ群の機能を実現するツールである．

さらに重要なことは，変更可能な変換

DDIF集合体 (DDIF Aggregates) の一覧表			
ARC	円弧	LS1	特別レイアウト
BEZ	ベジエ曲線	LSD	線種
CRF	コンテント参照	LW1	ラップ属性
CTD	コンテントの定義	OCC	頻度
CTS	コンテントのスタイル	PDG	ページ記述
DDF	文書のルート	PGL	ページレイアウト
DHD	文書のヘッダ	PGS	ページセレクト
DSC	文書についての情報	PHD	パス定義
EOF	セグメントの終わり	PTD	パターン定義
EXT	外部コンテント	PTH	合成パス
ERF	外部参照	PVT	プライベートコンテント
FAS	塗り潰しコンテント	RCD	レコード定義
FTD	フォントの定義	RGB	イメージルックアップテーブル
GLA	ギャラリ（フレーム）	SEG	ドキュメントセグメント
GLY	ギャラリ・レイアウト	SFT	ソフトダイレクティブ
GTX	一般テキストコンテント	SFV	ソフト値ダイレクティブ
HRD	ハードダイレクティブ	SGA	セグメント属性
IDU	イメージデータユニット	SGB	セグメント拘束
IMG	イメージコンテント	TBS	タブ設定
LG1	汎用レイアウト	TRN	変形
LIN	連線直線	TXT	LATIN 1 テキストコンテント
LL1	レイアウト属性	TYD	タイプ定義

表1　DDIF集合体一覧表

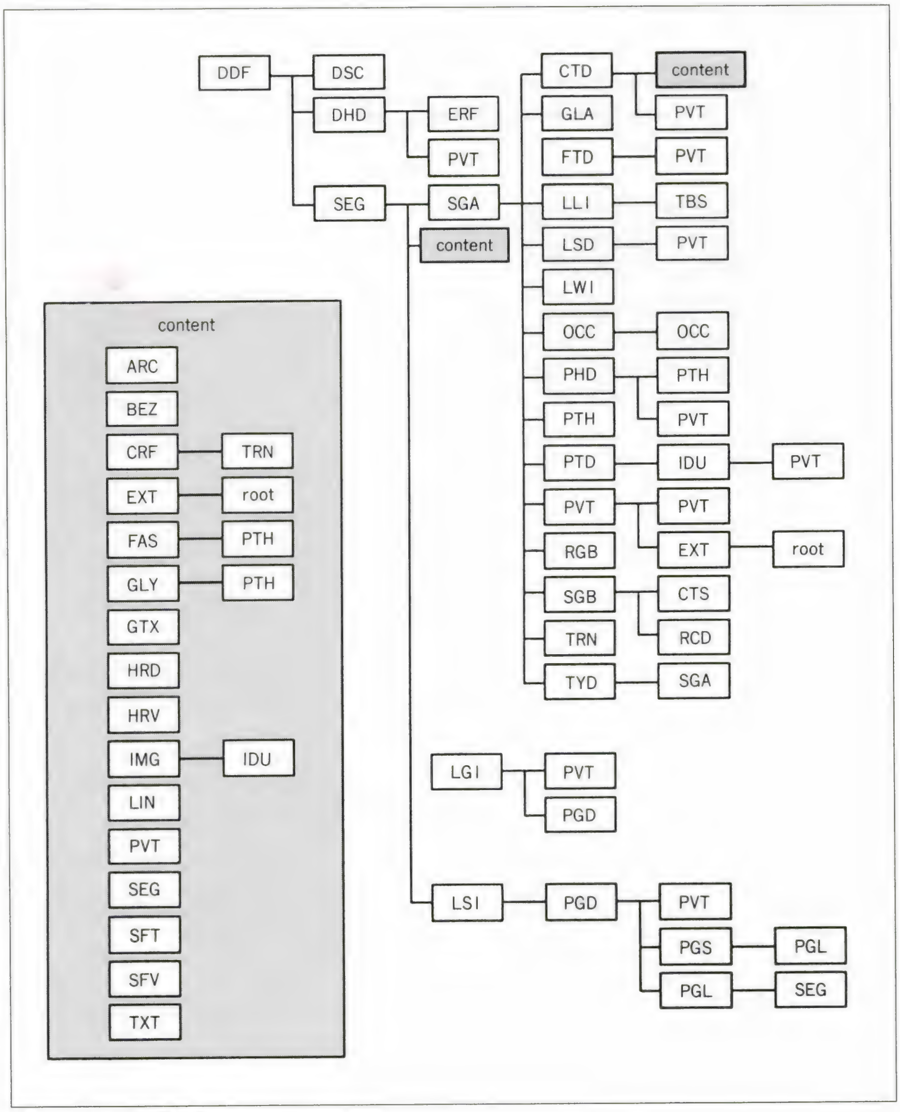


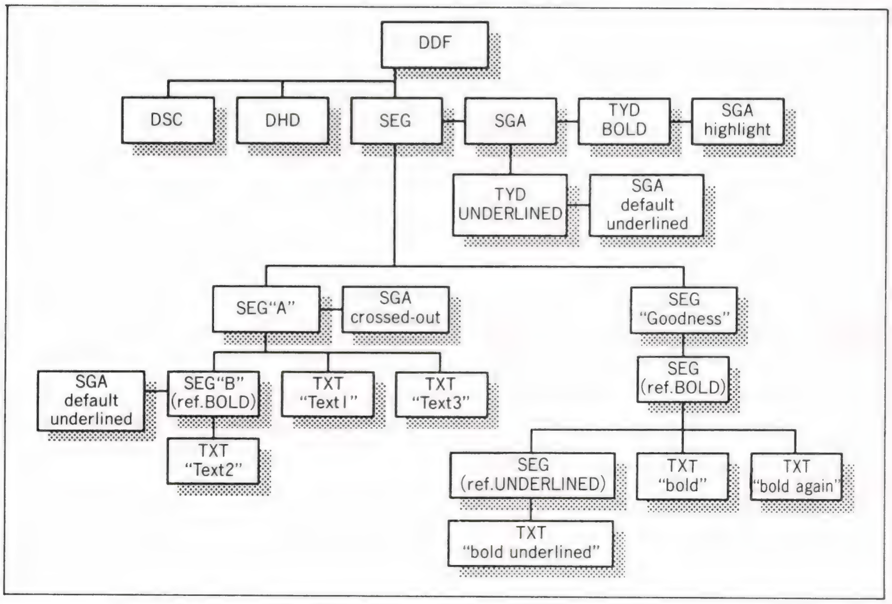
図2　DDIFの階層構造

図 3　サンプルドキュメントとDDIFリスト

注）このリストは，解析バックエンドコンバータ（DDIFを読んで，サンプルのような出力をするコンバータ）の出力であり，DDIFをCDAツールキットを用いて作成する際に，デバッグや作られるファイルの解析に用いられる．

サンプルドキュメント

bold <u>bold underlined</u> bold again




```
DDIF_DOCUMENT
{
 DDIF_DESCRIPTOR
 {
  DSC_MAJOR_VERSION 1
  DSC_MINOR_VERSION 0
  DSC_PRODUCT_IDENTIFIER "DDIF$"
  DSC_PRODUCT_NAME
  (
   ISO_LATIN1
"Hand-generated Standard DDIF Example"
  )
 }
 DDF-HEADER
 {
  DHD_VERSION
  (
   ISO_LATIN1  "V1.0"
  )
 }
 DDF_CONTENT
 {
  SEG_SPECIFIC_ATTRIBUTES
  {
   SGA_TYPE_DEFNS
   {
    TYD_LABEL  "BOLD"
    TYD_ATTRIBUTES
    {
     SGA_TXT-RENDITION
     (
      RND_HIGHLIGHT
     )
    }
   }
   {
    TYD_LABEL  "UNDERLINED"
    TYD_ATTRIBUTES
    {
     SGA_TXT_RENDITION
     (
      RND_UNDERLINE
     )
    }
   }
  }
  SEG_CONTENT
  {
   SEG_ID  "A"
   SEG_SPECIFIC_ATTRIBUTES
   {
    SGA_TXT-RENDITION
    (
     RND_CROSS_OUT
    )
   }
   SEG_CONTENT
   {
    TXT_CONTENT  "Text1"
   }
   {
    SEG_ID  "B"
    SEG_SEGMENT_TYPE  "BOLD"
    SEG_SPECIFIC_ATTRIBUTES
    {
     SGA_TXT_RENDITION
     (
      RND_DEFAULT
      RND_UNDERLINE
     )
    }
    SEG_CONTENT
    {
     TXT_CONTENT  "Text2"
    }
   }
   {
    TXT_CONTENT  "Text3"
   }
  }
  {
   SEG_ID  "Goodness"
   SEG_CONTENT
   {
    SEG_SEGMENT_TYPE  "BOLD"
    SEG_CONTENT
    {
     TXT_CONTENT "bold "
    }
    {
     SEG_SEGMENT_TYPE  "UNDERLINED"
     SEG_CONTENT
     {
      TXT_CONTENT  "bold underlined "
     }
    }
    {
     TXT_CONTENT  "bold again"
    }
   }
  }
 }
}
```

フォーマット（DDIFとDTIF）が持っているメモリ内フォーマットの機能は，このツールキットにより実現される．

また，このツールキットはCDAの中核となるDDIFのファイルを表示する機能を持っており，OSに組み込まれているCDAビューア（CDA Viewer）の基礎ともなっている．

LiveLink機能

LiveLinkサービスは，CDAのアプリケーション群内で，各種アプリケーションの統合をサポートする機能であり，現在のところ，複合文書エディタであるDECwriteとDECdecisionに組み込まれている．

◇CDAの強力な機能LiveLink

ACS（Application Control Services）は，コンバータライブラリとともに，マルチベンダー環境におけるアプリケーション間のゲートウェイの機能を提供する．ACSにより，CDAの変換フォーマットDDIF，DTIFを，データの格納フォーマットとしてサポートするアプリケーションは，LiveLinkの機能を使い，データをほかのアプリケーションと併用することができる．

LiveLinkは，ネットワーク上のほかのシステムでほかのユーザーにより作られ，処理され，格納されているデータを，自分のデスクトップ装置上のアプリケーションの中に取り込み，つねに最新のデータを自分のアプリケーションのデータとして保持することを可能にする．つまり，自分の取り込んだデータが，その処理されているシステムで更新されると，自分のアプリケーション上のデータも自動的

“変更可能”な文書と“最終形式の文書”について

変更可能な文書

（Revisable Format Document）

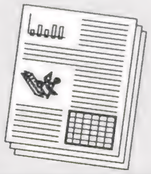
このフォーマットでは，文書の内容とともに，文書の最終フォーマットを作成するときに使われるパラメータや，改行や改ページなどを指定するダイレクティブ（指示命令）を記述している．これらのパラメータやダイレクティブは，文書の構成要素の抽象的な関連性を記述し，文書の最終的な表記の仕方を決定するときに用いられる．変更可能な文書フォーマットにおいては，決定されたページ番号，章番号，また，索引といったものは記述されない．これらの要素は，文書の最終形式において作成されるが，その作成を制御するパラメータという形で，記述されている．また，変更可能な文書フォーマットでは，文書の内容の最終形式における文書上の割り振りは記述されない．ページの割り振りの基本的なテンプレート（型）と文書の内容の最終形式での割り振り方を制御するパラメータという形で記述される．

最終形式の文書

（Final Format Document）

最終形式の文書フォーマットにおいては，すべてのフォーマッティング（ハイフンの挿入，改行，改ページなど）の決定がされ，計算の結果によって決定されるすべての文字要素（ページ番号，章番号，目次，索引など）が挿入されている．また，外部参照として記述されているすべての文書の内容も取り込まれている．

端的にいえば，最終形式は，ディスプレイや，プリンタ用紙などに表示するときのフォーマットで，章とか段落といった概念は持っていない．これに対して，変更可能な形式では，こういった論理的な関係を持たせることができるので，たとえば，途中に章を追加・削除したとき，その後ろの章番号などを自動的に変更することができる．



DTIF集合体

（DDIF Aggregates）の一覧表

CAT	コラム属性
CLD	セルデータ
CLR	セル
COR	コラム
DSC	文書についての情報
DTF	文書のルート
ERF	外部参照
EXP	数式
FMI	フォーマット属性
LPT	言語優先テーブル
HDR	文書ヘッダ
NMR	名前つき領域
NVL	プライベートデータ
RNG	領域
RWR	行
TBL	テーブルセグメント
TMD	テーブルメタデータ
WND	ウィンドウ

表2　DTIFの集合体一覧表

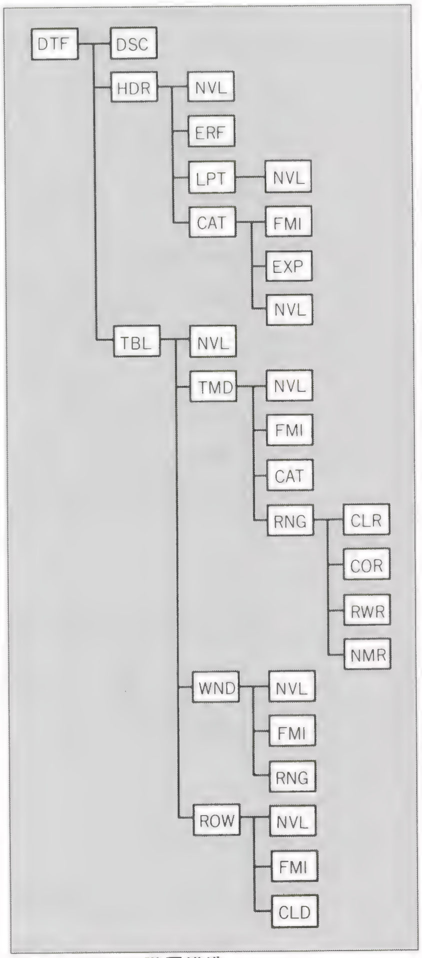
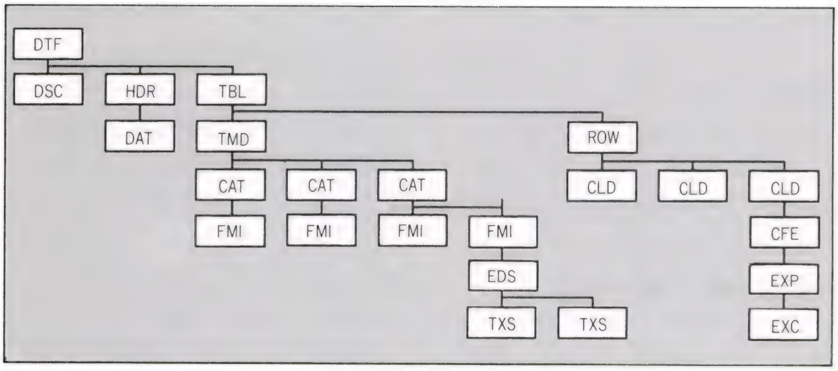


図4　DTIFの階層構造



サンプルテーブル

1983	Direct Labor	$200,000.00

```
DTIF_DOCUMENT
 {
  DTF_DESCRIPTOR
   {
    DSC_MAJOR_VERSION  1
 ! Longword Integer
    DSC_MINOR_VERSION  0
 ! Longword Integer
    DSC_PRODUCT_IDENTIFIER
 "%H44544624"  ! Byte String = "DTFS"
    DSC_PRODUCT_NAME
     (
       ISO_LATIN1
 "DTIF Sample Application V1.0"
     )
    DSC_ENCODE_MAJOR_VERSION  1
 ! Longword Integer
    DSC_ENCODE_MINOR_VERSION  0
 ! Longword Integer
   }
  DTF_HEADER
   {
    HDR_TITLE
     (
      ISO_LATIN1 "DTIF EXAMPLE"
     )
    HDR_DATE
     {
      DAT_DATETIME
 "%H135908021133393A"
 ! Byte String = ".Y...39:"
     }
   }
  DTF_TABLES
   {
    TBL_MAX_COLS  3   ! Longword Integer
    TBL_MAX_ROWS  7   ! Longword Integer
    TBL_METADATA
     {
      TMD_FLAGS
 "%B00000100000000000000000000000000"
 ! Flags  [Default value.]
      TMD_COLUMNS
       {
        CAT_ID  1       ! Longword Integer
        CAT_FORMATS
         {
          FMI_ALIGNMENT ALIGN_CENTER
 ! Integer = 1
         }
       }
   {
    CAT_ID  2  ! Longword Integer
    CAT_FORMATS
     {
      FMI_WIDTH  16 ! Longword Integer
     }
   }
   {
    CAT_ID  3  ! Longword Integer
    CAT_FORMATS
     {
      FMI_WIDTH  17 ! Longword Integer
     }
      {
       FMI_C_NUMERIC_FORMAT    ! Integer = 0
       FMI_NUM_DATATYPE
 "%B10000000000000000000000000000000"
 ! Flags [Default Value.]
       FMI_NUM_C_NUM_EDITSTR    ! Integer = 1
       FMI_NUMDES_EDITSTR
        {
         EDS_EDIT_STRING_C
          (
           EDS_MINUS_LITERAL   ! Integer = 27
           EDS_CURRENCY        ! Integer = 14
           EDS_CURRENCY        ! Integer = 14
           EDS_DIGIT_SEP       ! Integer = 5
           EDS_CURRENCY        ! Integer = 14
           EDS_CURRENCY        ! Integer = 14
           EDS_CURRENCY        ! Integer = 14
           EDS_DIGIT_SEP       ! Integer = 5
           EDS_CURRENCY        ! Integer = 14
           EDS_CURRENCY        ! Integer = 14
           EDS_CURRENCY        ! Integer = 14
           EDS_DIGIT_SEP       ! Integer = 5
           EDS_CURRENCY        ! Integer = 14
           EDS_CURRENCY        ! Integer = 14
           EDS_DECIMAL_DIGIT   ! Integer = 7
           EDS_RADIX_POINT     ! Integer = 8
           EDS_DECINAL_DIGIT   ! Integer = 7
           EDS_DECINAL_DIGIT   ! Integer = 7
           EDS_MINUS_LITERAL   ! Integer = 27
          )
         EDS_EDIT_STRING
          (
           {
            TXS_TEXT_STRING
             (
              ISO_LATIN1  "("
             )
           }
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            0  ! Integer
            {
             TXS_TEXT_STRING
              (
               ISO_LATIN1  ")"
              )
            }
          )
        }
      }
     }
   }
  TBL_ROWS
   CLD_STATE  CS_ISVALUE        ! Integer
   CLD_VALUE_C CV_LATIN1_TEXT   ! Integer
   CLD_VALUE "%H31393833"
 ! Byte String = "1983"
  }
  {
   CLD_STATE CS_ISVALUE       ! Integer = 0
    CLD_VALUE_C CV_LATIN1_TEXT   ! Integer = 1
    CLD_VALUE "%h446972656374204C61626F72"
 ! Byte string = "Direct Labor"
   }
   {
    CLD_STATE CS_ISVALUE      ! Integer = 0
    CLD_VALUE CV_INTEGER      ! Integer = 0
    CLD_VALUE "%V200000"
 ! 3-byte Variable Integer
    CLD_FORMULA_CFE
   {
    EXP_LIST
     {
      EXL_EXPR_C LIT_INTEGER  ! Integer = 0
      EXL_EXPR 200000    ! Longword Integer
       }
     }
    }
   }
  }
 }
```

図5　サンプルドキュメントとDTIFリスト

注）このリストは，解析バックエンドコンバータ（DTIFを読んで，サンプルのような出力をするコンバータ）の出力であり，DTIFをCDAツールキットを用いて作成する際に，デバッグや作られるファイルの解析に用いられる．

に更新される．

LiveLinkの目的は，

・マルチベンダー環境における，個々のアプリケーションを統合する．

・アプリケーション間の対話における方式，標準仕様，そして通信仕様などのモデルを提供する．

◇CDAフォーマットとLiveLink

交換フォーマット（DDIF，DTIF）は，複合文書の中に，LiveLinkのデータを包含できるように設計されており，LiveLinkコネクションで統合されるアプリケーション群によってサポートされるデータ交換の標準フォーマットとして使用されている．DECは，ACSV 1 .0のLiveLinkの機能をDECwriteとDECdecisionに組み込みの形で出荷した．

◇3種類のLiveLink

データリンク：CDA文書から他のファイルへのポインタを持たせて，データを文書内に取り込む機能である．これにより，ネットワーク上のデータ（テキスト，図形，イメージ）を変更可能なフォーマットの間（最終形式に変更されるまでの），そのファイルへのポインタという形で保持し，データの冗長性を極度に低減させている．同時にそれらのデータについては，つねに最新のデータであることを保証する．

アプリケーションリンク：ほかのアプリケーションにより作成され，編集されているCDA交換フォーマットの文書内のデータを自分のアプリケーションの中に取り込むときに用いられる．それと同時に，取り込まれる外部データの処理をする外部アプリケーションへのポインタも取り込み，自分のアプリケーションからポインタ先のアプリケーションを起動することができる．

スクリプトリンク：複数のアプリケーションを使って反復処理の結果としてのデータと，その反復の手順（スクリプト）を自分の複合文書に取り込むときに用いられる．このリンクは，DECdecision／Builderによって前もって登録された命令群を含む処理手順を実行する．

DECwriteにおいてユーザーは，Link

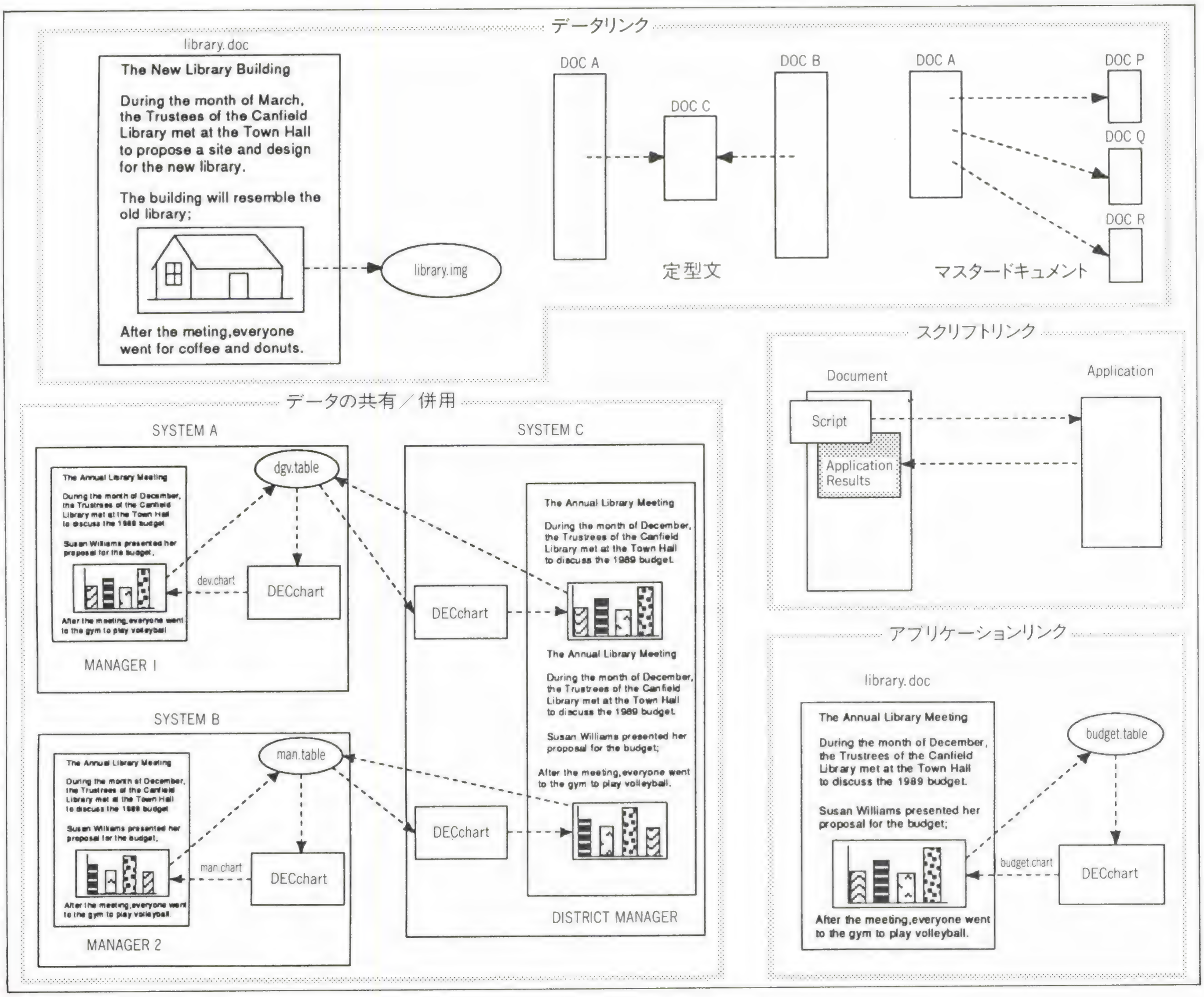


図6　LiveLink

Menuで単にLiveLinkを選択するだけでLiveLink機能を起動できる．このMenuからユーザーは，DECwriteから，ほかの複合文書，イメージ，オンライン上にある図形情報，DECchart内のChart，DEC decisionでは，Calcのスプレッドシート，Accessのデータベース，Builderの処理手順（スクリプト）等々へのリンクを簡単に設定できる．

OSの機能強化

VMS（VAXのOS）においては，レコード管理サービス（RMS――Record Management Service）の機能を強化し，CDAのデータを通常の文字列に自動的に変換する機能を付加した．この機能強化によって現存する非CDAアプリケーションからもCDAアプリケーションで作られたデータを適切に読むことができる．

このOSの機能強化によって，文字，図形，イメージなどを含む複合文書を，従来から使われている文字端末ででも，限られた図形処理機能しか持っていない端末ででも，その端末で表示できる限りの情報を表示することができる．

ユーティリティの機能強化

メールやOS上のプリントユーティリティは，CDAのファイルフォーマットを自動的に認識し，適切な処理をする．また，DECwindows Mailにおいては，受信したメールがCDAのデータであることを認識すると，適切なCDAビューアを起動させる．

CDAベースのアプリケーション群

DECが開発したCDAの機能を取り込んだキーアプリケーションは，DECwriteと，DECdecisionである．

DECでは，CDA対応ソフトウェア開発者に対して，全面的サポートをしている．その一例として，開発アプリケーションからCDA交換フォーマットやコンバータに簡単にアクセスできるCDAツールキットを提供している．

CDAと国際業界標準

前述のように，CDAの開発に当たり，変更可能な文書交換フォーマットの基礎には，ISOによる標準であるASN．1に準拠するDDISを開発した．また，最終形式文書交換フォーマットの基礎には，SPDLの決定までの間，デファクトの標準であるPostScriptを採用した．

◇DDIS採用の主な利点

・ISOおよびCCITT（Consultative Committee on Telophony and Telegraphy）の標準に適合する．

・他のネットワークやデータプロトコルでサポートされているコード化標準を使える．

・コード自体がコンパクトになり，しかもデコードが容易である．

・拡張性がある．

・バイナリを含む取り込みデータのシンタックスを容易に扱える．

DDIFとSGMLおよびODA/ODIF

DDIFは，現存するStandard Generalized Markup Language（SGML，ISO 8879）と，採択されつつあるISO Office Document Architecture and Interchange Format（ODA/ODIF, ISO 8613）に対しては，高度な互換性を持つように設計されている．マルチベンダー環境にあるユーザーのためには，できうる限りの業界標準に準拠する必要がある．また，ビジネスが最新技術のコミュニケーションを駆使して地球規模で行なわれている現在では，さまざまな国の種々雑多なコンピュータを統合した地球規模のシステムにおける情報処理をスムーズに遂行しなければならないというニーズもある．この対策の１つとして，単純にSGMLかODA／ODIFを採択し，DDIFのデータフォーマットをその拡張版あるいは特殊版として定義することもできる．しかし，ユーザーの切実な課題である国際業界標準への準拠という明確なメリットも慎重な検討の結果，次のようなデメリットを考え，SGMLもODA／ODIFも採択しなかった．

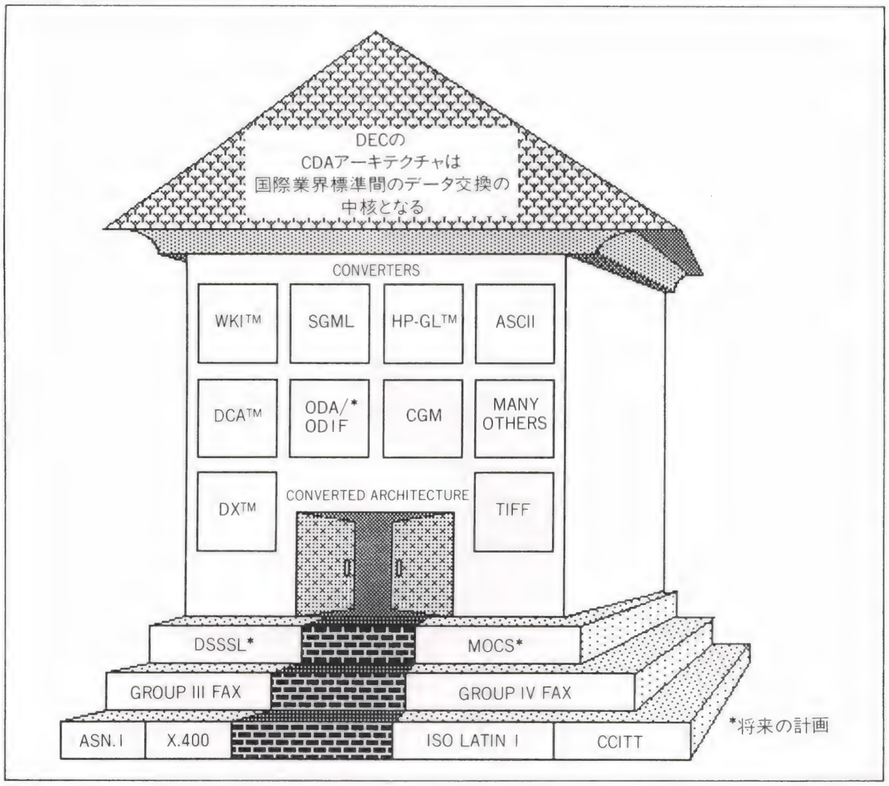


図7

（SGMLのデメリット）

・SGMLとODA／ODIFの間には，互換性がない．

・SGMLはデリミッタベースであるばかりでなく，SGML要素とASN．1要素の混在が不可能である．

・SGMLにおいては，文字情報以外のデータは，標準的には，外部参照データとしてしか取り込めない．

・SGML標準は，テキストのデータ属性，レイアウト機構，図形やイメージの標準という構成要素を定義しないが，DDIFはこれらを必要とする．

（ODA／ODIFのデメリット）

・ODAは，CDAにとって核ともいうべき項目において不適合であった．

というのは，ODAを採択すると，必要とされるODAの拡張機能は，でき上がるCDAをODAに準拠できないものに変えてしまう．たとえば，複数の文書間やアプリケーション間のリンケージを処理する機能は，CDAにおいては不可欠だが，ODAではオプションの拡張機能として追加しなければならない，など．

・ODAは，文字情報と非文字情報のデータタイプを統合できない．ODAで使われている手法は，現存する統合されていない標準群のデータのレイヤ上に，文書構造の構成要素のレイヤを取り入れて複合文書フォーマットとしているにすぎない．

・DECは，汎用および特殊文書構造の処理方法をも含めて，ODAの文書モデルの動向を注意深く見守り，CDAのODAに対する互換性を保持していく．しかし，“文書の構造要素と文書の内容要素とは，ODAにおいて処理されているよりも，より高度に統合されるべきである”という方針も維持していく．

DDIFとほかの国際業界標準

これに関しては，以下のように要約できる．

a. DDIFのテキスト要素は，ISOの標準文字コードを直接取り込む．

b. 図形要素は，一般的に広く使われているANSI／ISOのGKS（Graphical Kernel System）と互換性を保つ．

c. 文書本体のイメージは，PostScriptと互換性を保つ．

d. イメージ要素は，CCITTのバイトーナル（FAX）エンコーディング標準を，カラーイメージをサポートする機能拡張をも含めて採用している．

e. テーブルデータについては．種々の業界標準とも呼ぶべきものが存在している．そこで，DTIFにおいては，すべての重要と思われる業界標準フォーマットで使われているすべての機能のための１種類のコード化標準を定義している．

DTIFは，１つの製品に特有であり，ほかの製品とのデータの交換において，あきらかに適切でないと思われる機能については，標準的な処理機能を定義していないが，このようなケースに対して，DTIFには，プライベートエンコーディング（特殊要素のコード化）のための汎用の回避機構を提供している．

f. DOTSシンタックスが対処する問題に対する国際業界標準は，現在のところ存在しないが，CCITTのX.400メール／メッセージ標準が，DOTSが提供する複合オブジェクト転送機能（Multiple Object Transport Function）に最も近い標準である．したがって，DECは，DOTSシステムのコード化を，複合オブジェクトのメッセージ交換の際のX.400標準の処理方法とできるだけ互換性が持てるように設計した．

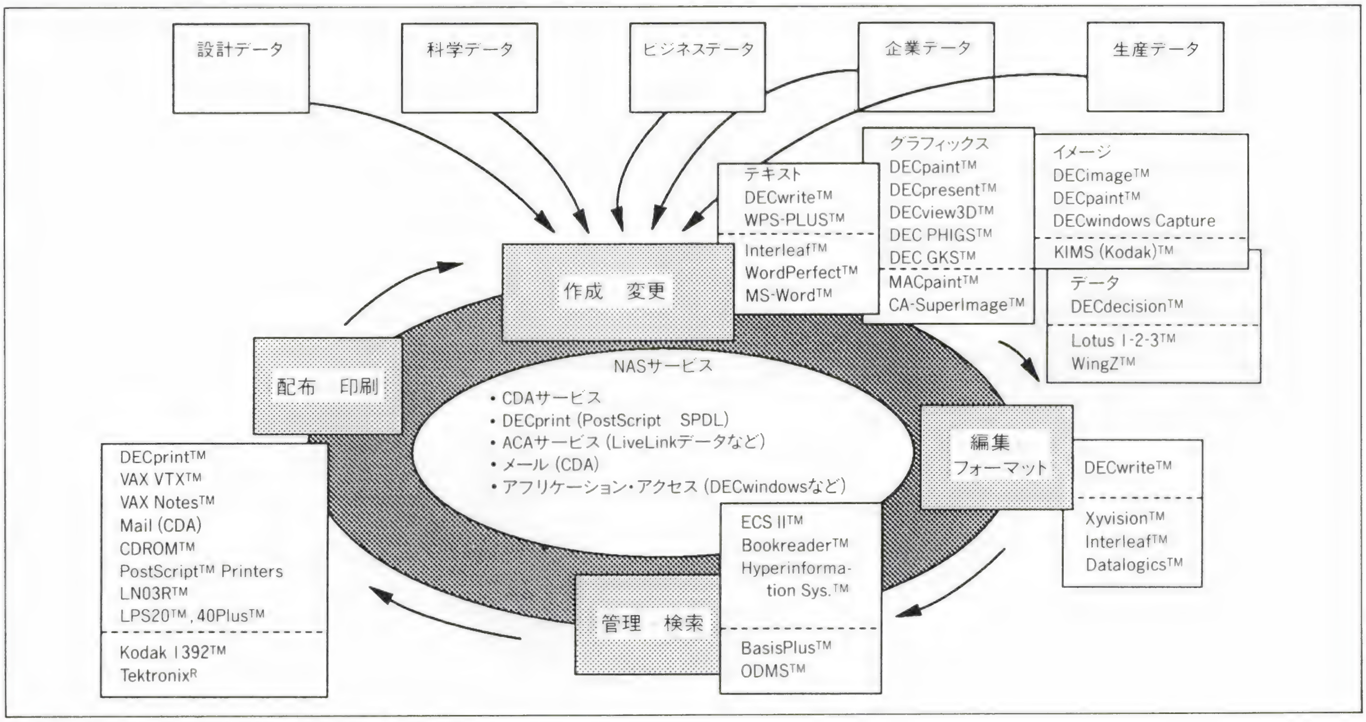


図８　DECの提供するCDAによる統合された複合文書処理

キーワード

フォントの表現フォーマット その2

リコー中央研究所 技師長 村山 登

活字に始まった印刷用フォントは，写植機のフィルムフォントから，コンピュータ用デジタルフォントへと変化しつつある．ここでは，これらフォントの表現フォーマットとして，最も実績のあるIKフォーマットと，最新のINSTACフォーマットを取り上げた．IKフォーマットについては，前号からの続きをURWの大伴氏が，INSTACフォーマットについてを筆者が本号で紹介する．

IKフォーマット

URW大伴研究室 大伴武都美

前号で，イカルス（“IK”はイカルスの略称）システムの概要を紹介し，イカルスにおけるアートワーク，デジタル化，フォントのデータベースなどについて述べた．今号ではそれに続くイカルスシステムフォーマット（IKフォーマット）の内容を示し，完結としたい．

デザイン変換機能

すべての変換はデータベースフォーマットであるIKフォーマットを基礎にして行なわれる．デザイン変換機能も例外ではない．

以下にごく簡単にイカルスがサポートするモディフィケーション・プログラムのいくつかを記す．

CN： 等高線付け
IT： イタリック化
UI： 平体化
SK： ステム幅の変換
RD： 丸み付け
AN： 古典化処理
SH： 影付け
HL： 陰線処理
GX： Xハイトの変化
IN： インターポーレーション

インターポーレーションは，日本でもすっかり有名になってしまったが，書体ファミリーを増やすのに役立つ．これはたとえば，ライトとボールドの書体を入力することによりミディアムの書体を自動的に作成する，という作業に使われる．これはただ単に中間の線幅の書体を自動作成するだけでなく，パラメータの設定によりさまざまなバリエーションの中間の書体を作成できる．

マシンフォーマット

まず，**図1 a）～c）**を見ていただきたい．これはマシンフォーマットの説明のために好んで使われる図である．

マシンフォーマットは，データベース

図1 a)カーブの記述法と自由度

関数名	自由度
ピクセル	0
ランレングス	1
ベクトル	2
アーク	3
2次スプライン	4
g-コニック	5
ベジエ（3次スプライン）	6

b)カーブの記述法と必要メモリ

		必要メモリ	
		per 100×100 char	per 1000×1000 char
0	ピクセル	1200 bytes	120.000 bytes
1	ランレングス	450 bytes	4.500 bytes
2	ベクトル	60 bytes	300 bytes
3	アーク	300- (2 bytes per point)	600 bytes (4 bytes per point)
4	2次スプライン	125- (2 bytes per point)	250 bytes (4 bytes per point)
5	g-コニック	105- (2 bytes per point)	210 bytes (4 bytes per point)
6	3次スプライン	100- (2 bytes per point)	200 bytes (4 bytes per point)
	IK	200 bytes	

c)カーブの記述法とスピード

		スピード		
		低解像度	高解像度	comparison
0	ピクセル	最高速		
1	ランレングス	高速	高速	
2	ベクトル	良	高速	
3	アーク	やや良	非常に高速	
4	2次スプライン	低	良	1.6
5	g-コニック	低	良	1.3
6	3次スプライン	低	良	1.0
7	IK	各マシンへのコンバートが必要		

フォーマットとはおのずから異なった必要性があり，小さいメモリで高速にきれいな文字を描かねばならない．そのためマシンフォーマットは次のような特徴を備えている．

●カーブの記述法と自由度

曲線の自由度は明らかに次数が高くなるに従って増加する．1個のピクセルでは1個のポイントしか描けないが，1つのベジエ曲線は，長いスパンを描くことができる．

●カーブの記述法と必要メモリ

自由度が大きくなると必然的に必要メモリは減少する．

エム角(ボディサイズ)100×100と1000×1000の場合について比較し，参考のためにIKフォーマットの必要メモリも記載してある．円弧以上の自由度の曲線については，100×100の場合，2bytes／ポイント1000×1000の場合，4bytes／ポイントを割り当てるとする．もちろんベジエが一番メモリが少なくなるが，2次スプラインやコニックに比べてそれほどの効果はない．

●カーブの記述法とスピード

これは曲線の自由度に反比例して働く．IKについては，これはデータベースのフォーマットであるので，マシンフォーマットにはかなわない．

◆　◆　◆

大事なことは，それぞれの機器に最適のマシンフォーマットを選ぶことである．2400lpi(line／inch)のタイプセッタと240lpiのプリンタが同じフォーマットを持たねばならないという必然性は何もない．もっと重要なことは，さまざまな最適マシンフォーマットに容易にコンバートできるようなデータベースにしておくことで，IKフォーマットは，それを可能にする．

使用可能なアウトラインフォーマットの一部を例示する．

DI	ディスプレイ（直線と円弧）
BE	ベジエ（**図2**）
CN	コニック（**図3**）
VC	線分と円弧(ただしDIと異なる)
VE	線分
QS	2次スプライン（**図4**）

ラスタフォーマット

イカルスのシステムにとっては，ラスタフォーマット（ビットマップ化）もフォーマット変換の1つとして考えられ，システマチックに行なわれる．もう一度**図3**を参照していただきたい．IKデータはコンバータを通して，ランレングスフォーマット（SC Format）に直接変換するほか，いわゆるヒント付きデータベースで，これからさまざまなフォーマットに変換してNimbusや他のRIPに送ることができる．URW社としてはプリンタ用に公開されたフォーマット（Tre Type Formatなど)への変換コンバータを供給していく予定である．（完）

謝辞
本稿内のイカルスフォーマットの説明に使用したサンプル文字は，株式会社タイプバンクのご好意によるものであり，ここに謝意を表します（筆者）．

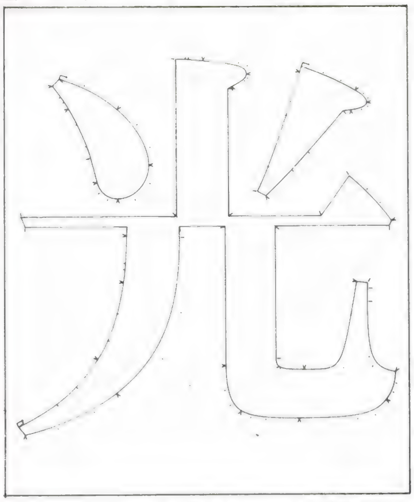
注）　さらに詳しく知りたい方に

イカルスあるいはデジタルフォントについてのさらに詳しいことは，
ピーター・カロウ著　「書体のデジタル・フォーマット」　定価1万円（送料込み）
に述べられている．本書は下記に直接申し込む（送り先等必要事項をファックスで送る）と入手できる．

札幌市東区北33条東13丁目3-43-1001
URW大伴研究室　大伴武都美
☎：011-752-8010
Fax：011-752-8011

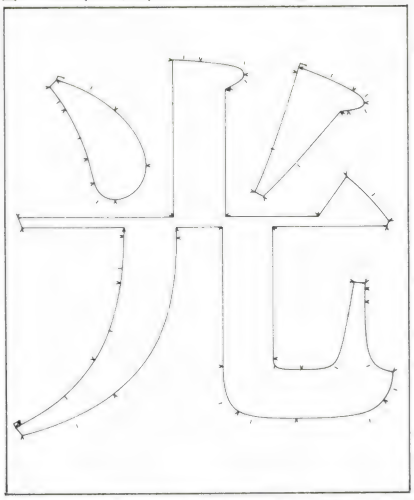
ビジネスについての問い合わせは，
URW Unternehmensberatung Karow Rubow Weber GmbH Harksheider Strasse 102D-2000 Hamburg 65West Germany
☎：001 49 40 60 60 52 25(日本語使用)
Fax：001 49 40 60 60 51 48（日本担当マネージャー　大澤）

図2　BE（ベジエ）フォーマットへの変換の例



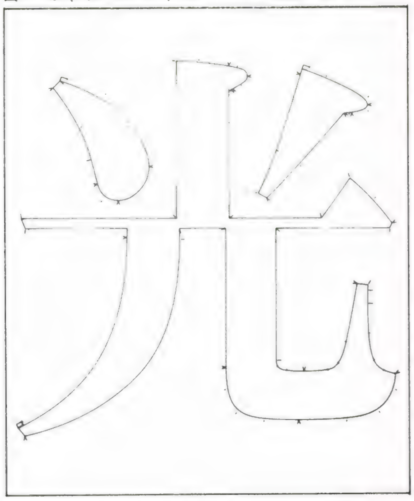
（©タイプバンク　明朝ヘビィ）

図3　CN（コニック）フォーマットへの変換の例



（©タイプバンク　明朝ヘビィ）

図4　QS(2次スプライン)フォーマットへの変換の例



（©タイプバンク　明朝ヘビィ）

INSTACフォントフォーマット

リコー中央研究所　村山　登

INSTACフォーマットとは

この特徴ある和製フォーマットの説明に入る前に，制定までの経過を述べる必要があるだろう．

日本規格協会の情報技術標準化研究センター（INSTAC）の委嘱により，高品質フォントの標準化調査研究委員会（長谷部紀元委員長）が，調査研究の成果としてまとめたのがこのフォーマットである．詳しくは，高品質フォントの標準化調査研究報告書（平成元年，平成2年）を参照されたい．筆者はこの委員会の一員として，またWG 2（エンコードと実現技術のワーキンググループ）の主査として調査を担当してきたが，委員各位の衆知の結晶がこのフォーマットであり，問題点はあるものの，現在におけるもっとも汎用で先端的なアウトラインフォントフォーマットであると思う．

したがって，このフォーマットがそれ以前にどこかで使用されていたという実績があったわけではないが，その後日本規格協会関連の文字フォント開発・普及センターが発足し，フォントの開発，登録，流通のセンターとして機能して以来，このセンターの開発するフォントにこのINSTACフォーマットが使用されることになっている．

具体的には，センターの公募に当選した明朝体とゴシック体のフォントがそれぞれ，このフォーマットに準拠した点属性付きの，中心点円弧曲線とB-スプライン曲線のフォーマットになって流通する予定である．

つまり，このフォーマットは，どの特定の企業とも直接の関係はなく，その意味では初めてのオープンフォーマットである．その特徴は，

1）バイナリフォーマットだけではなく，テキストフォーマットも持っているのでポータビリティ（可搬性）がよい．

2）ほとんどあらゆるアウトライン（輪郭線）表現方式に対応できるので，欧文文字，日本語文字などすべてを包含できる．

3）オープンである．

一方，その裏返しに次のような問題点もある．

1）データ量が多く冗長な部分がある．

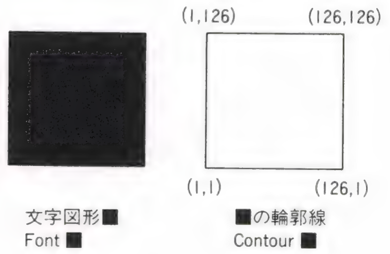
2）どんな方式にも対応できる半面，どんな方式とも直接の互換性はなく変換が必要である．

3）オープンではあるが使用実績に乏しい．

このような特徴から，このフォーマットは，フォントのリソース（ソースフォント）の交換フォーマットとして最適なものであり，実際の機器に搭載する際にはオブジェクトフォントフォーマットに変換して搭載するのが望ましい．また，報告書（前記）にも明記してあるが，このフォーマットが日本の標準になったわけではなく，参考フォーマットとして作成されたものであり，ISOにも正式な標準案としてではなく，参考フォーマットとして提出されていて，ISO9541に反映できるかどうかは今後の課題である．

ソースフォントの性格上，機器固有のメトリック情報やスケーリング，ヒンティング情報などを考慮していないので，オブジェクトフォントにして機器に搭載する際は，これらを追加する必要がある．

図1　文字図形“■”の輪郭先記述例（ライン）




```
LNo.
 1  #
 2  #        テキスト形式フォントファイルの例 1
 3  #
 4  (MASTER DESIGN  (DESIGN ID              01)
 5                  (FAMILY ID              01)
 6                  (VERSION NO             1.1)
 7                  (SIGNIFICANT BIT      0 x 22)
 8                  (BODY GRID SIZE         128)
 9                  (COORDINATE           RIGHT)
10                  (PRODUCTION DATE 11171988)
11                  (DESIGN NAME          "例題")
12                  (COPYRIGHT            "高品質フォントの標準化調査研究会")
13                  (CHR SET              ISOMULTI94 0 x 42))
14
15  (CHAR 0 x 23
16      (CONTOUR
17          (INTERNAL)                        # 内包閉曲線なし
18          (LIN (1 1)  (126 1))              # 下の水平線
19          (LIN (126 1)  (126 126))          # 右の垂線
20          (LIN (126 126)  (1 126))          # 上の垂線
21          (LIN (1 126)  (1 1))))            # 左の垂線
22
23  (END OF DESIGN)
```

L1-3：# 以降はコメント文である．論理的意味を持たない．
L4-13：ヘッダ情報であり，デザイン番号，コピーライトなどを示す．
L7：0x 22は文字符号の上位が16進で22であることを示す．
L8：文字のボディサイズが128であることを示す．
L9：表現している座標系が右手系，つまり反時計まわりの座標系であることを示す．つまり，X軸を親指方向にとるとY軸が中指の方向となり，左下が基準点(0,0)である．
L13：ISOの94文字単位のマルチバイトコード系の一つの0x 42(0100 0010)の文字セットであることを示す．
L15-21：文字図形の輪郭線情報である．
L15：下位バイトが16進の23であることを示す．
L16：輪郭線記述の開始．
L17：輪郭線の内部には輪郭線がないことを示す．
L18-L21：輪郭線が直線であることを示す．
L21：文字図形記述の終了を示すトレーラーである．

アウトラインフォント フォーマットの概念

アウトラインフォントのフォーマットは次の3つの部分から構成される．

1）ヘッダー
2）輪郭線記述
3）トレーラー

1）は著作権などのフォントの履歴に関する情報である．

2）はフォントを構成する字形(Glyph)の論理的，数学的記述で，アウトラインフォントでは輪郭線の記述であり，フォーマット全体の大部分を占める．

3）は終わりを示す情報，脚注，訂正，参考文献などで少量である．

具体例として，中黒の四角形の文字(■)の例で説明する．

INSTACフォーマットでは，フォントデータをすべて文字で表現できる．もちろん，よりコンパクトなバイナリフォーマットも併記しているが，ここではテキストフォーマットにしぼって説明する．

図2は同じ文字を線分（Line）ではなく，折れ線(Polyline)で表現した場合のリストであり，L18以降がコンパクトに表現されている．輪郭線表現に必要な情報をまとめると次の3つである．

曲線種と特徴点列

この例のように，LINとかPLNは輪郭線の一部を表現する曲線種であり，その曲線種を表現するために必要な点列を曲線の特徴点（Feature Point）と呼ぶ．輪郭線の記述にはこの曲線種とその特徴点列の座標値を記述すれば指定された曲線が生成できることになる．

点属性

ほかの曲線への変換を容易にするために，補助的に特徴点において曲線が滑らかに接続されているか（TAN)，とがって接続されているか(EDG)，特徴点が同一曲線上の点であるか（CRV）の，接続点属性を記述することもある．

輪郭線ループとループ回転方向

図1，2の“■”では，輪郭線ループが2重だが，“回”では輪郭線ループが4重になっている．このような場合に，記述しようとする輪郭線ループが内側から何番目のものかを指定する必要があり，これが（INTERNAL）である．

また，輪郭線ループが黒をまわるループ（内側）と白をまわるループ（外側）で回転方向を逆にしておいたほうが輪郭線間の塗り潰し（ペイント）のとき便利である．もし，左から右に走査するとすれば，**図1**，**図2**の例では，下向きの輪郭線が黒の始まりで（各図右側の座標で(1，126）から（1，1）へ)，上向きの輪郭線は白の始まり（同じく座標(126，1)から（126，126）へ）である．またたとえば，“□”のように，内側にもう1つの輪郭線ループがあれば，内側ループの回転方向を逆にすれば，外ループも内ループも，下向きは黒の始まり，上向きは白の始まりとなり，ペイントが高速にできる．

図2　文字図形“■”の輪郭先記述例（折れ線）

```
LNo.
 1 #
 2 #        テキスト形式フォントファイルの例 2
 3 #
 4 (MASTER DESIGN  (DESIGN ID              01)
 5                 (FAMILY ID              01)
 6                 (VERSION NO             1.1)
 7                 (SIGNIFICANT BIT      0×22)
 8                 (BODY GRID SIZE        128)
 9                 (COORDINATE          RIGHT)
10                 (PRODUCTION DATE 11171988)
11                 (DESIGN NAME        "例題")
12                 (COPYRIGHT          "高品質フォントの標準化調査研究会")
13                 (CHR SET            ISOMULTI94 0×42))
14
15 (CHAR 0×23
16       (CONTOUR
17           (INTERNAL)                         #内包閉曲線なし
18           (PLN (1 1)  (126 1)  (126 126)
19                (1 126)  (1 1))))
20
21 (END_OF_DESIGN)
```

輪郭線表現方式の種類

図1のLIN（Line）とか**図2**のPLN（Polyline）は輪郭線の論理的表現形式であるがINSTACフォーマットでは全部で11種類の曲線表現方式を選定している．これを互換性の見地から大分類すれば次の4通りとなる．

1）直線（L）
2）円弧，楕円弧（C）
3）パラメトリック多項式曲線（P）
4）点属性指定，線種無指定（N）

選定した11種類の曲線方式を**表1**に示す．

図1や**図2**のLINやPLNのところに，この曲線種記号が使用されるのである．詳しくは報告書に譲るとして，曲線表現の要点だけ説明する．

表1　表現方式

表現方式	記号	大分類
1）線分(Line)	LIN	L
2）折れ線(polyline)	PLN	L
3）3点円弧(3 point Arc)	ARC_3	C
4）中心点円弧(Center Arc)	ARC_C	C
5）楕円弧(Ellipse)	ELP	C
6）2次スプライン(Conic Spline)	SP2	P
7）3次スプライン(Cubic Spline)	SP3	P
8）2次ベジエ(Conic Bezier)	BZ2	P
9）3次ベジエ(Cubic Bezier)	BZ3	P
10）拡張ベジエ(Extended Bezier)	BZE	P
11）線種無指定(Not Specify)	NONE	N

円弧とパラメトリック多項式曲線の互換性

上記4つの大分類のC（円弧）とP（多項式曲線）は代表的な曲線表現形式であるが，この両者の間での互換性はどうやってとればよいのだろうか．

欧文フォントでは伝統的に円弧表現をよく使用してきたが，最近もっともよく使用されるのは**図3**,**図4**に示すような大分類のPのパラメトリック多項式である．

たとえばTEXのMetaFontやPostScript-PDLのType 1は3次ベジエ曲線を採用しているし，またTrueImage-PDLのTrueTypeではコニックベジエ曲線を採用している．

図3,**図4**のパラメトリック多項式の数式表現には次の3つの形式があるが，生成される曲線はまったく同一であり，完全に互換性がある．

なお**図3**はベジエ曲線，**図4**はベジエ曲線を接続したB-スプライン（BasicSpline）であり，特徴点列Piを結ぶ折れ線を，ベジエの多角形（BezierPolygon）と呼ぶ．

1）ベジエ表現（**図3**;1-3）

$$B=\sum_{i=0}^{n}{}_nC_i(1-t)^{n-i}t^iP_i$$

B：成点，${}_nC_i$：2項係数，

t：媒介変数；$0\leqq t\leqq 1$，P_i：特徴点

2）拡張ベジエ表現（**図3**-4）

$$BE=\sum_{i=0}^{3}C_i(1-t)^{3-i}t^i$$

$C_0=P_0, C_i=\{cP_0-(c-3)P_0\}$,

$C_3=P_3, C_2=\{dP_3-(d-3)P_3\}$

c, d：ふくらみ係数

3）スプライン表現（**図4**）

図4の曲線を・のところで分割すれば次の表現に変換できる．

$$S=\sum_{i=0}^{n}A_i(1-t)^{n-i}t^i$$

A_i：係数ベクトル（P_iの線型結合）

しかし，残念ながらこれらのパラメトリック多項式曲線と円弧表現の間には，厳密な数学的互換性はないので，互換性をとる方法には次の3つの方法がある．

1）有理多項式を使用する．
2）曲線を分割する．
3）多重多項式を使用する．

1）はもっともオーソドックスな方法であり，数学的にも完全に互換性がとれる．しかし，補正多項式で割る割り算が必要であり，高速生成には向いていない．ここでは，高速生成が可能な2）と3）の方法を述べる．

図5は3次ベジエ曲線で半円を近似したものであるが，半円との半径誤差は最大2％もあり，とても半円とはいえない．しかし，**図6**のように1/4円に分割して生成すれば誤差は2桁減少する．

3）の方法は拡張ベジエ曲線のふくらみ係数c, dをたとえば

$$c=d=3.5+2(1-t)t$$

と多重としたものであり，その半径誤差は5/10000よりも小さく，ふくらみ係数の次数を上げると誤差はいくらでも小さくできる（**図7**）．

接続点属性指定，線種無指定，互換表

大分類のN（None）は輪郭線の種類は指定せず，輪郭線と輪郭線の接続点の属性のみを記述したものである．つまり，折れ線表現の点が，

1）同一曲線上の点（Curve=CRV）であるか
2）異なる曲線間の尖点（Edge=EDG）であるか
3）異なる曲線間の滑点（Tangent=TAN）であるか，

図4　B-スプライン曲線

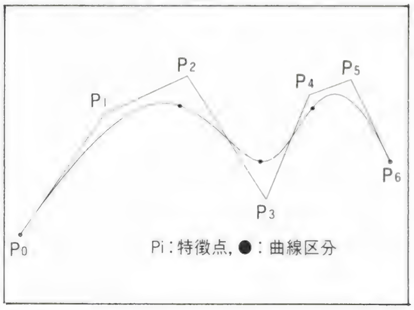


図3　ベジエ曲線と拡張ベジエ曲線

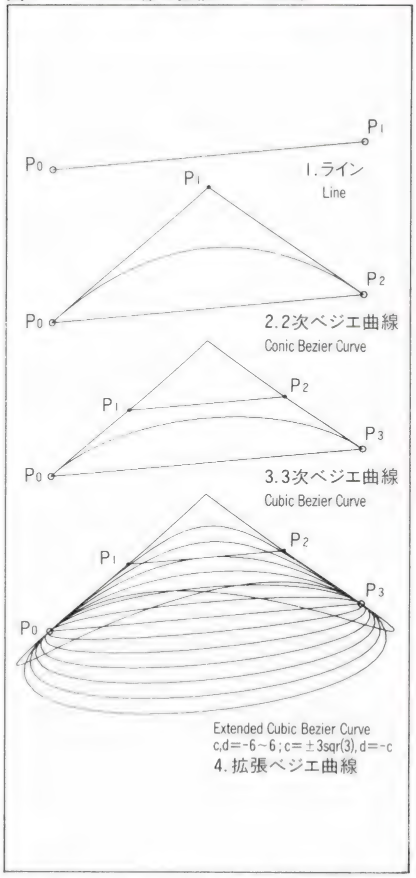


図5　ベジエ曲線による半円生成（等間隔媒介変数）

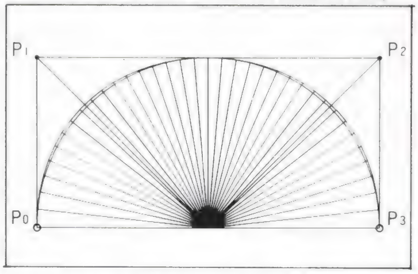


図6　ベジエ曲線による1/4円生成（等間隔媒介変数）

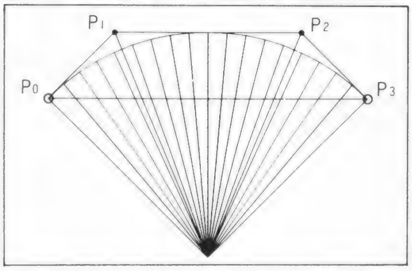


図7　多重多項式による半円生成（等角度媒介変数）

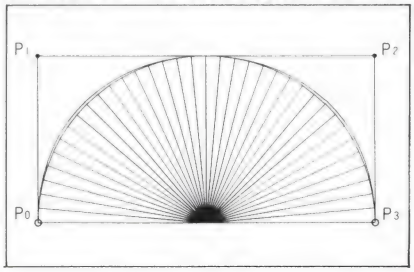


表2　互換表

	LIN	PLN	ARC3	ARCC	ELP	SP2	SP3	BZ2	BZ3	BZE	NON
LIN	---	=									
PLN	=	---									
ARC3	=	=	---	=	=						
ARCC	=	=	=	---	=						
ELP	=	=	=	=	---						
SP2	=	=	A	A	A	---		=			
SP3	=	=	A	A	A	=	---	=	=	=	
BZ2	=	=	A	A	A	=		---			
BZ3	=	=	A	A	A	=	=	=	---	=	
BZE	=	=	A	A	A	=	=	=	=	---	
NON	=	=	P	P	P	P	P	P	P	P	---
	---：恒等，=：完全互換，A：近似互換，P：処理後互換										

を記述しておいて，あとで任意の曲線表現に容易に変換できるようにしたものであり，IKフォーマットとの互換性を考慮したものである．

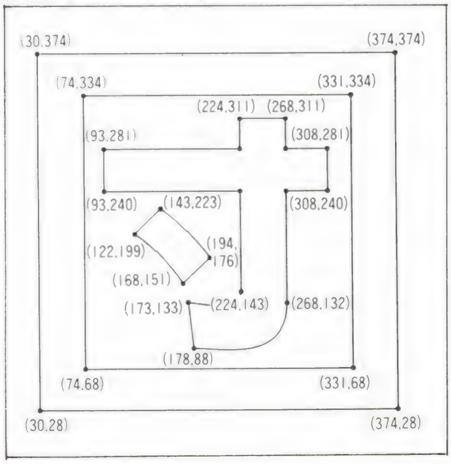
以上をまとめて示したものが**表2**である．

複雑な例

図8，**図9**に複雑な記述の例として“団”(句点コード3536)の記述例を示す．この中で使用されているSP3は3次スプライン曲線を示すが，SP3のところの最初から2番目と最後から2番目の値はそれぞれ始点，終点における接線の方向を示すベクトル値であり，それ以外の値は曲線上の点列の座標値である．この点列は**図4**のベジエ多角形の点列によるB-スプライン曲線の特徴点列のように，曲線からはずれた点列ではない．

また，SP2，SP3，B-スプライン曲線などは，与えられた点列や方向ベクトルから，それらの線型結合である係数ベクトルAiに変換する変換式が必要である．

図8　“団”（3536）のアウトライン



おわりに

以上INSTACフォーマットの要点を説明した．このフォーマットは今までのどの方式も包含できるように考慮してあるので，ソースフォントのフォーマットとして最適なものである．しかし，機器に搭載するオブジェクトフォントに変換する際に冗長性を排除し，スケーリング，ヒンティング情報などの情報や，機器固有のメトリック情報を追加して使用するようにしなければならない．コントリビューターの一人としてオープンフォーマットであるこのフォーマットが，フォントの流通に寄与できることを心から希っている．

また，冗長性などの問題点の検討や，標準化案として，より洗練されたフォーマットにするのが今後の課題である．

参考文献

1）日本規格協会:高品質フォントの標準化調査研究報告書（1989, 1990）．
2）村山，江成:DTP用フォント，画像電子学会誌，Vol.18, No.3, pp.97-107（1989）．
3）N.Murayama:A Polynomial for Automatic Contour Vector Coding, Trans. IEICE Vol.E72, No.5, pp.661-669（1989）．
4）村山:曲線生成のための多重多項式，情報処理学会論文誌，Vol.31, No.8, pp.1213-1220（1990）．

図9　“団”のリスト

```
    .
    .
)  End of Header
(CHAR 68 # First Character Data 3536
    .
    .
(Contour Negative # Sense of rotation=
 clockwise
(PLN(30 374)(374 374)(374 28)(30 28)
 (30 374))
# Begin of Internal List #1 Contour
(Internal
(Contour Negative # Begin of Internal
 List #2 Contour
(Internal
(Contour Negative
(PLN(224 281)(224 311)(268 311)(268 281) (308 281)(308 240)(268 240)(268 132))
(SP3(268 132)(0 -1)(264 110)(249 94)
 (221 87)(1 0)(178 88))
(LIN(178 88)(173 133))
(SP3(173 133)(25 -3)(198 131)(214 131)
 (221 134)(0 -97)(224 143))
(PLN(224 143)(224 240)(93 240)(93 281)
 (224 261))
(Internal) #No Internal Contour
(Contour Negative
(SP3(147 223)(27 -15)(174 210)(182 192)
 (-12 18)(194 176))
(LIN(194 176)(168 151))
(SP3(168 151)(-10 13)(158 164)(139 184)
 (15 -15)(122 199))
(LIN(122 199)(147 223))
(Internal) #No Internal Contour
) ) #End of Internal List #2 Contour
(PLN(74 334)(331 334)(331 68)(74 68)
 (74 334))
) ) ) )
```

MIPS

MIscellaneous Packet Stream

パソコン1024夜

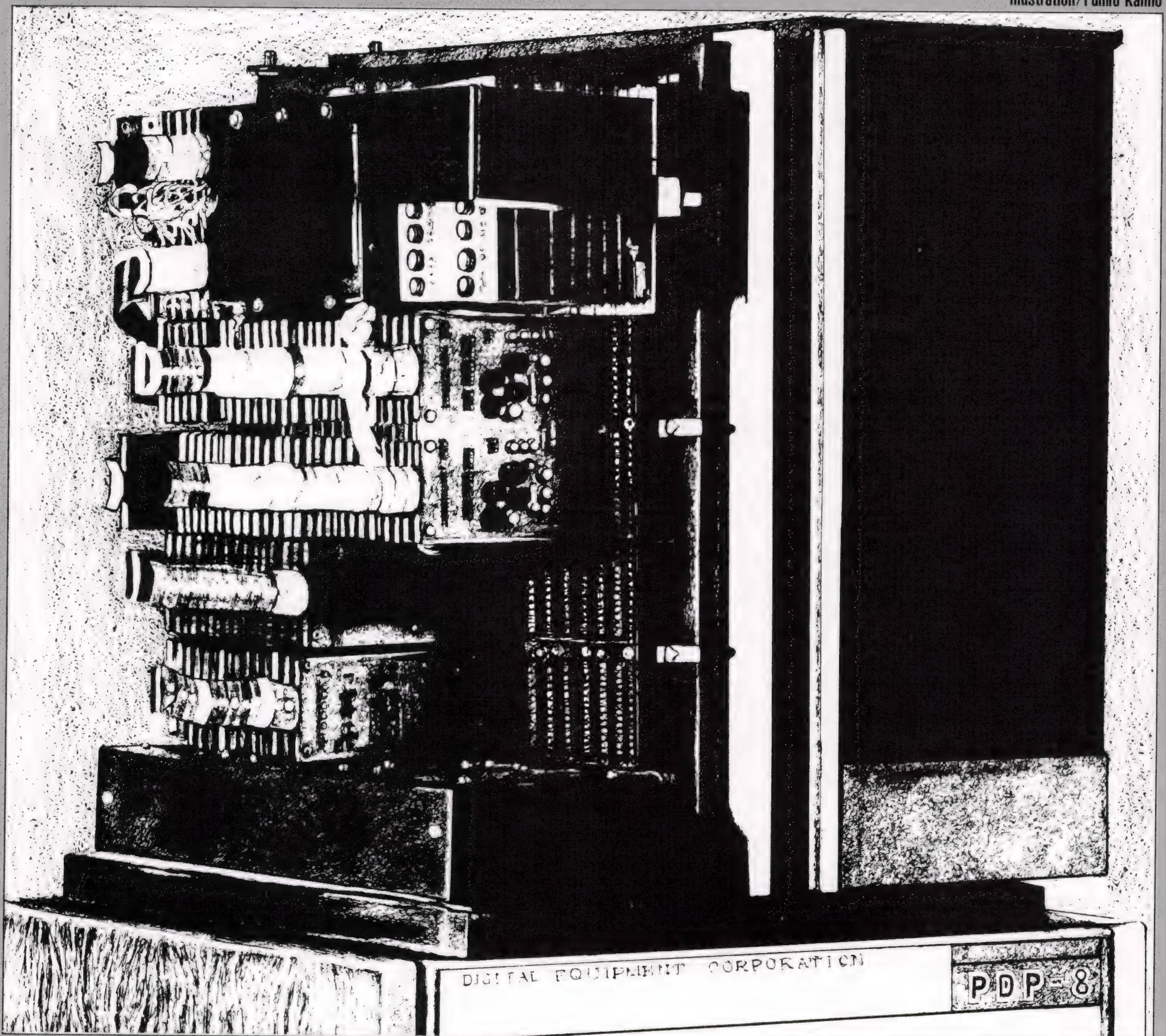
プログラマのための数学的世界観入門

ウサギと月と，そして太陽と

Fat Bits

BOOK SHELF

illustration/Fumio Kanno



1966年８月に発表された，DEC(Degital Equipment Corporation)のPDP-８．ここからPDP-11，さらにVAX11へと続くミニコンピュータの時代が始まった．

第4夜　カツアゲ的不安について

福崎俊博

中学生の天敵

私は地図が好きだ．道路地図だろうと住宅地図だろうと，地図を眺めていると数時間くらいあっという間にたってしまう．たまたま知らない場所に行ったり，タクシーで初めての道を通ったりすると，家に帰ってから地図を広げて，自分はこの道を通ったんだなあと，しみじみ思うのである．もともと活動的なほうではなかったが，ここ何年かの私の行動範囲は特に狭く，我ながらまるで小学生並みだと思う．知らない土地を歩くのは好きなのに，旅行は大嫌いという厄介な性格のせいもあって，地図を眺めるという屈折した趣味に走っているのかもしれない．

それでも中学生や高校生の頃は，あちこちに出かけていったものだ．大体，親と一緒にいたくない年頃なので，学校の帰りに友だちと夜まで遊んでいたり，休日は繁華街をぶらついたりしていた．高校生になると行動範囲はさらに広がったが，それには学校に行きたくなかったし家にも帰りたくなかったという事情があって，あちこちで遊んでいても，それほど楽しいと感じていたわけではない．

それはともかく，中学生の頃，学校の帰りに遊ぶというのは，今にして思えばほとんどアドベンチャーゲームの世界だった．街は大きなダンジョンで，学生服姿の我々の行く手には，警官やPTA，補導員などといった魔物たちが待ち受けているのである（アドベンチャーゲームと違うのは，さすがにこうした魔物たちを打ち倒して何かを奪ったりはしなかったという点だ）．

しかし，悪いことは何もせずに，ただふらふら歩いているのが取り柄の我々にとって，真の天敵は他校の中学や高校に通うツッパリの方々なのだ．ツッパリが1人でいるときは何でもないが，2人以上のツッパリに遭遇したら，発見される前にそれとなく脇道にそれなければならない．目と目が合ったが最後，「ガンを飛ばした」とか「因縁を付けた」とかいう無茶な理由でベコベコに殴られるか，金品をまきあげられるに決まっているのだ．もっとも，大した理由もなくケンカしたり，校舎や体育館の裏に呼び出しをかけたりするのは日常茶飯事だったから，別にツッパリの人々だけが特殊な思想の持ち主だったわけではなかったようにも思う．我々は，金品の強奪をしなかっただけの話だ．

そんなわけで，中学時代には，そこら辺をのらのらするだけでもけっこうスリルがあった．実際，友だちに付き合ってもらって病院に行く途中，高校生くらいのツッパリ3人組に取り囲まれたことがある．「おい，お前ら，金持ってるだろ」「金？　金なんかねえよ」———弱いくせに向こうっ気だけは強い（一番やられやすい最悪の性格）のは今も昔も変わらないが，当時，陸上部で鍛えた逃げ足には自信があった．「身体検査してもいいんだな」「いいよ」———ポケットの中には病院で支払う予定の千円札があったのでまずいと思ったが，殴られそうになったら逃げればいいやと考えていた．ところがツッパリの皆さんは，意外にも「分かったよ，いっていいよ」とか何とかつぶやいて，私たちを許してくれたのである．

私はがっかりしてしまった．なんていうお間抜けなツッパリさんたちだろう．第一，裏道に連れ込むわけでもなく，道路のまん中でカツアゲに及ぶなど，カツアゲの基本がまるで分かっていない．私は自分の身の安全を喜ぶのも忘れ，勝手に腹を立てながら友だちと病院に向かったのであった．

インストールプログラムは嫌い

大人になってからは，さすがにツッパリに取り囲まれる不安を感じることはなくなった．殴られるとかカツアゲされるとかいう具体的な不安感よりも，大人になってからはむしろ，私の場合には漠然とした不安のほうが強いように思う．自分が会社とかの組織に属していないという不安，生活や将来に関する不安など，あえて自分で解説すればそういうことになるのかもしれないが，実際にはもっとぼんやりとした，とらえどころのない不安なのだ．まあ，そうした不安があるからこそ，せきたてられるように仕事に打ち込めるといえなくもないのだが．

もちろん，そういう自分でも分かりに

くい不安以外に，日常生活上の不安はある．コンピュータユーザーとしてハードディスクのクラッシュや停電は勘弁してもらいたいし，キーボードにはジュース類の缶を近づけないようにしている．そういう種類の不安なら，コンピュータを使う以上はまあ仕方ないかなと納得できるが，どうにもたまらないのが，アプリケーションをインストールするときの不安感なのだ．

パーソナルコンピュータを使い始めて10年以上になるが，いつだってソフトウェアを買ってきたときは嬉しい．早く使ってみたい気持ちを抑えながら，パッケージを開けてマニュアルを読む．これが一番楽しい作業だ．

ところがである．マニュアルを読んでも，インストールの方法がよく分からない場合が多いのだ．いや，「分からない」というのは正確ではない．たいてい，付属の「インストールプログラム」というのを実行するようにマニュアルに書いてあるから，インストールプログラムを実行すれば何かしてくれて，アプリケーションを使えるようになるのだな，とは思う．

しかし，私はどうもこのインストールプログラムというのが好きになれない．中学時代のカツアゲ同様，自分の意思と無関係に勝手に何かされてしまうというのがとてもつらいのである．ユーザーの指定したディレクトリにファイルをコピーしてくれるだけならよいのだが，勝手にディレクトリを作ったり，CONFIG.SYSやAUTOEXEC.BATを書き換えたりとか，やりたい放題のインストールプログラムもあるからだ．

AUTOEXEC.BATなどのファイルをインストールプログラムが書き換えたり新規に作ったりする場合には，たいていバックアップを取ってくれるようになっている．だが，プログラムの質問に間違えて答えて，「STOPキーで中断できます」ということなので中断してそのままやり直したりすると，バックアップされたファイルに上書きする形でもう1度バックアップが行なわれて，自分で作った元のファイルがなくなってしまうケースがあるのだ．気が付くと100行はあったはずの中身がたったの2行になっていたりして，こういうときにはかなりがっかりする．

一方，中には傑作なインストールプログラムもあって，昔の「直子の代筆」では，数分かかるインストールの実行中に，面白いクイズを出題してくれた．この抱腹絶倒のインストールプログラムは空前であると同時に絶後でもあったようで，インストールが簡単になった最近の「直子の代筆」では残念ながら廃止されてしまったようだ（PC-9801版での話）．インストールプログラムもこのくらい洒落っ気があれば，私でも少しは好きになれるかもしれない．

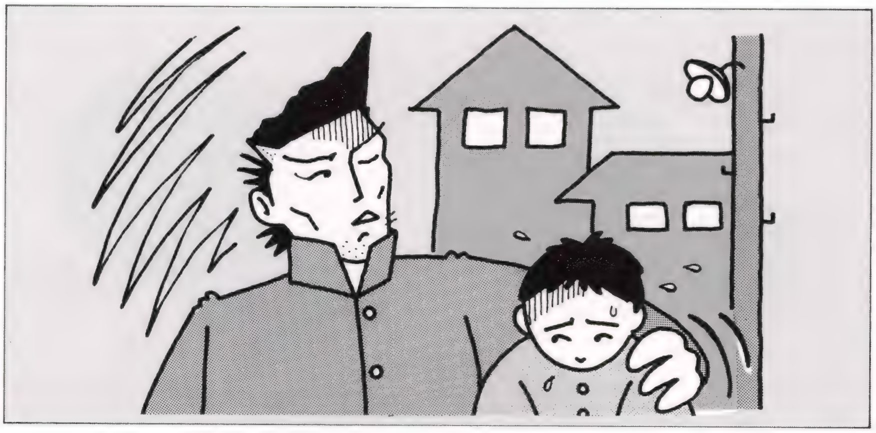
詳しい地図さえあればいい

Windowsや最近のMS-Cの場合など，インストール時に何かファイルを生成しなければならないようなケースでは，インストールプログラムの存在意義も理解できる（出来がよければ）．でも，それ以外のソフトウェアの場合にはどうだろうか．もちろん，初心者用にオマケとして付けるのは親切だろう．ただ，何をされるのか分からないというカツアゲ暴力沙汰的な不安を感じる私のような人間のために，インストールプログラムに頼らなくても手動でインストールできる方法をマニュアルに書いておいてもらえると，たいへんありがたいと思う（もちろん，きちんと書いてあるマニュアルも多い）．

マニュアルは地図みたいなものだから，初心者が迷子にならないようにしなくてはいけないとは思うが，その一方で詳細な情報も書いておいてくれないと，私などはいつ裏道に引きずり込まれて殴られるかと不安になってしまう．アプリケーションの実行にどのような設定が必要なのか，オリジナルのフロッピーに収められたファイルのうち，どれが最低限必要でどれがオマケなのかということや，環境設定用のファイルがあるなら，パラメータの詳細な内容についてもマニュアルに書いてあると嬉しい（ついでにアプリケーション独自のファイル形式や，データの内部表現などについても解説されていればもっとよい）．ソフトウェアメーカーには，インストールプログラム嫌いの変人の心情も何とかご理解願いたいと思う．

O氏の恐怖体験

このコラム担当の編集者O氏も，かつてカツアゲされた経験があるそうだ．有り金全部渡しても，まだあるだろうと納得しないツッパリさんがO氏のポケットをまさぐると，お札に似た感触の紙があった．「これは何だよ」と怒ったツッパリさんが取り出した紙は何と模擬試験の成績表．ツッパリさんはO氏の成績を見ると，「いいじゃねえか」と感心して去っていったという．ツッパリにも，けっこう面白いやつが多いのである．



プログラマのための

数学的世界観入門

データの誤りの訂正や検出をするためには、誤りの程度を評価するための基準が必要になる。

斎藤 明

1 ハミング距離と符号の強さ

さて前回，符号とはどのようなものかその概略を説明した．nビットのビット列を送るときそのうちのtビットまでの誤りならば訂正できるような符号をt-誤り訂正符号，同様にtビットまでの誤りならば検出できるような符号をt-誤り検出符号とよぶ．また，2つの符号$x=x_1x_2...x_n$，$y=y_1y_2...y_n$ (x_i，$y_j=0$ or $1, 1\leq i,j\leq n$)のハミング距離$d(x,y)$は

$$d(x,y)=|\{i \mid 1\leq i\leq n,\ x_i\neq y_i\}|,$$

すなわち値の異なるビット位置の総数で定義されることは前回述べた．今回はまずハミング距離と上のtの値の関係を考えてみよう．

n桁のビット列全体のなす集合をYとしよう．Yは2^n個の要素を持つ．符号CはYの部分集合と考えられる．前回を読んだ人には$C=f(X)$である．前回述べたようにCはYの真部分集合であり，実際には$|C|$は$|Y|$に比べてかなり小さい．さて，Cの各要素xに対してxからのハミング距離がt以下であるようなYの要素全体のなす集合$B(x;t)$を考えてみよう．

ここで，次のようなイメージを考えてみると分かりやすい．今Yを平面にみたて，Yの各要素を平面上の点と思う．さてCの各要素xに対して半径tの円を描いてみよう（**図1**）．この円の周および内部が$B(x;t)$のイメージである．

(1) まずtが十分小さく，

$$x_1\neq x_2 \implies B(x_1;t)\cap B(x_2;t)=\phi$$

が成立している，すなわち$B(x;t)$たちに共通部分がないとしよう．平面のイメージでは各円が交わらない状況に対応する．このときCの要素xからハミング距離t以下にある点yはxを中心とする円の周上または内部にあり，しかも他の円に含まれることはない．したがってyをxに訂正できる（**図1**）．

(2) 次にtがもう少し大きくなって円たちの中に交わるものが現われたとしよう．ただし，各円は他の円の中心を含むほどには大きくないとする（**図2**）．今$B(x_1;t)\cap B(x_2;t)\neq\phi$であるとし，さらに$y\in B(x_1;t)\cap B(x_2;t)$としよう．このとき$B(x_1;t)$も$B(x_2;t)$も$x_1, x_2$以外に円の中心すなわち$C$の要素を含まないので，$y\notin C$であり，yが誤りであることは分かる．しかし$y$は$x_1, x_2$いずれからもハミング距離が$t$以下であるので，$y$を$x_1, x_2$いずれに訂正すべきであるかは分からない．すなわち誤りの検出はできるが訂正はできない．

(3) 最後にtがさらに大きくなり，円が他の円の中心を含んでしまったとしよう．たとえば$x_2\in B(x_1;t)$ ($x_1, x_2\in C$)としよう．このときx_1にtビット以下の誤りが生じてその結果がx_2となることがありうる．そして，もしそうなったとすると$x_2\in C$なので誤りが生じたことは分からない．すなわち誤りの検出すら保証されなくなる．

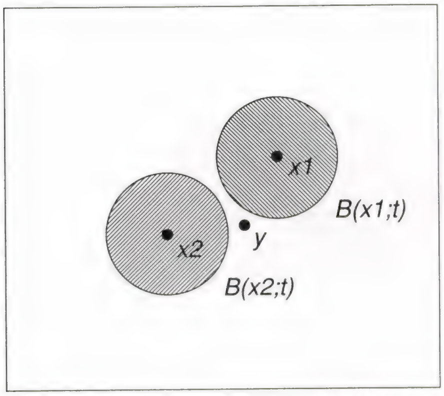


図1 すべてのxを中心としてハミング距離tの円を描く．交わらない場合には，円内の点yをxに訂正できる．

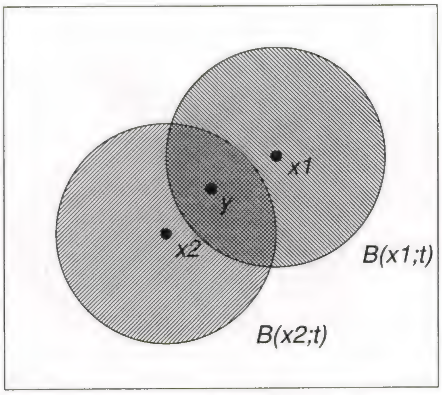


図2 tが大きくなって円同士が重なると，どちらの円に属しているか分からず，訂正できないyが現われる．

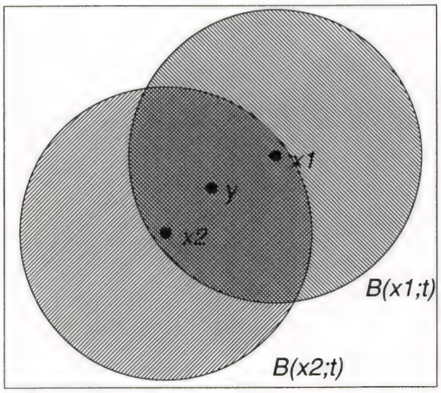


図3 円の中心が他の円に含まれるようになると，誤りが起きたことの検出もできないyが現われる．

さて符号Cの異なる2つの要素の中で最もハミング距離が近いものを考える．その距離をCの最小距離とよび，$d(C)$と表わす．

$$d(C)=\min\{d(x_1,x_2)\mid x_1,x_2\in C,x_1\neq x_2\}$$

このとき

(1) 各円が共通部分を持たないための必要十分条件は$d(C)>2t$. ただし$d(C)$は整数なので，これは$d(C)\geq 2t+1$と同値.

(2) 各円が他の円の中心を含まないための必要十分条件は（$d(C)$が整数である点も考えて)$d(C)\geq t+1$である.

以上より次の事実が分かる.

事実1　符号Cについて

(1) Cがt-誤り訂正符号であるための必要十分条件は$d(C)\geq 2t+1$

(2) Cがt-誤り検出符号であるための必要十分条件は$d(C)\geq t+1$ である.

この事実から$d(C)$が大きければ大きいほど符号の誤りの訂正，検出の力は強くなることが分かる.

2 反復符号

さて，ここで最も簡単な符号である反復符号を考えよう．反復符号は送るべきビット列の各ビットを何回か繰り返すものである. k回繰り返す反復符号をk連反復符号とよぶ. k連反復符号では送るべきデータ$x_1x_2...x_m$に対し符号

$$\underbrace{x_1x_1\ldots x_1}_{k個}\underbrace{x_2x_2\ldots x_2}_{k個}\underbrace{x_mx_m\ldots x_m}_{k個}$$

が対応する. この符号化関数をfとしよう. たとえば3連反復符号ではデータ010は$f(010)=000111000$に符号化される．今符号化する前のデータx,x'のハミング距離がlであるとすると$f(x)$と$f(x')$のハミング距離はklとなる．特に$l=1$ならば$d(f(x),f(x'))=k$，すなわち次の事実が成り立つ.

事実2　k連反復符号の最小距離はkである.

したがって，特にkが奇数ならばこれは$(k-1)/2$-誤り訂正符号となる. **プログラム1**に3ビットのデータに対して5連反復符号の符号化を行なうC言語によるプログラムを挙げておく.

3 伝送率

事実2により，反復符号では必要に応じて反復回数を大きくとることにより，最小距離をいくらでも大きくすることができる．すなわちいくらでも大きな誤り検出，訂正の能力を持つ符号を，反復符号で構成することができる．では，反復符号さえあれば他の符号は不要なのであろうか．実はそうではない．むしろ反復符号はあまりよくない符号であると考えられている．その理由を次に見てみよう.

```
#define LENGTH 3
#define REPEAT 5

unsigned int repeat(a)
unsigned char a;
{
        unsigned int b, c;
        int i, j;
        int k = LENGTH * REPEAT - 1;

        c = 0;
        for (i = 0; i < LENGTH; i++) {
                b = a & 01;
                for (j = 0; j < REPEAT; j++)
                  c = c | (b << k--);
                a >>= 1;
        }
        return(c);
}
```

プログラム1　5連反復符号化するプログラム．ただし，データはcharの下位3ビット（残りビットはすべて0），符号は16ビットのunsigned intの下位15ビットに入れておくことにしている.

たとえば7ビットのデータに対する2連反復符号を考えてみよう．この符号の最小距離は2であり，7ビットのデータは14ビットの符号に変換される．一方，次のような1ビットを付加する符号を考えてみよう.

$$f: b_1b_2...b_7\rightarrow b_1b_2...b_7p\ (b_i=0 \text{ or } 1,\ p=0 \text{ or } 1)$$

ただし,

$$p=\begin{cases}0\ \cdots\ b_1+b_2+\cdots+b_7が偶数のとき\\1\ \cdots\ b_1+b_2+\cdots+b_7が奇数のとき\end{cases}$$

とする(この符号をパリティ検査符号とよぶこともある)．するとこの符号の最小距離もやはり2となり，その誤り検出の力は2連反復符号と同じである．しかしこちらの符号長は8ビットであり2連反復符号の符号長より6ビット少ない．情報伝達に要する時間で比較すれば，2連反復符号ではデータそのものを送る場合に比べ2倍の時間がかかるのに対し，パリティ検査符号では8/7倍程度で済む.

今mビットのデータがnビットの符号に変換されるときn/mをこの符号の伝送率とよぶ. $n>m$なので伝送率は1より大きい．伝送率が大きいと符号化に伴う情報伝達の時間，コストが大きくなる．伝送率があまり大きくなっては，いくら最小距離が大きくとも実用上は役に立たない符号になってしまう．いかにして伝送率をあまり大きくしないで最小距離を大きくするかが符号理論の目的の1つである．また，誤りの検出をいかに簡単に行なうかということも重要である.

次回は一般的に使われている符号である巡回符号をテーマにする.

ウサギと月と，そして太陽と

第4回　世界はネットワークを待っている

砂原秀樹

ソニーの懸賞論文でもらったPalmTopがやっときました．いろいろ遊んでいるのですが，なかなか面白いですねえ（まあ，不満も多少ありますが……）．これで簡単なプログラミングもできたら，もっと面白いんだけどなぁ…….

実は，先日アメリカに行ったときもPalmTopを借りて持っていったんですが，UCB（カリフォルニア大学バークレイ校）のBSDのグループを訪ねたときに披露したところ，ゾロゾロと人が集まってきて，ちょっとした人気者でした．タッチペンと液晶パネルというPalmTopのインターフェイスに非常に興味を引かれたようです．特に，sendmail（TCP／IP上でのメール配送システム）の作者Eric Allmanは非常に気にいったのか，BSDのプロジェクトリーダーKirk McKusickに「買って!!」と言っていました．

最近，PalmTopに限らずタッチペンと液晶パネルというインターフェイスに興味を持っていて，ワークステーションにこういうユーザーインターフェイスを組み込むと面白いかなと思っています．とまあ，くだらない話はさておき，そろそろ本題に入りましょう．

★

前回は，導入されたワークステーションを使いやすくするためには，標準添付のソフトウェアだけでなく，世の中に流通しているさまざまなソフトウェア（売っているソフトということではなく，商品にしろフリーソフトウェアにしろ多くの人が便利だとして支持しているものという意味）を整備しなければならないことをお話ししました．しかし，ネットワークのないワークステーションなんて，「○×のないコーヒーのようなもの（ふっ，古い）」．今回は，ネットワークの構築についてお話ししましょう．といっても，技術的な話はあちこちでされているので，ちょっと毛色の違った話をします．

★

Unixというやつは，利用者が自由に設定できる代わりに，管理をきちんと行なわないと，すぐにスネてしまいます．ネットワークが，1研究室や1部署内でとどまっているうちは，管理を行なっているメンバーもたいてい気心の知れた者同士ですからまだよいのですが，ネットワークが大きくなると，「向こう側」の管理をやっている人の顔も知らないという状態も起こってきます．こうなってくると，話は面倒になってきます．

ネットワークを作るときは，ケーブルをどう敷設するか，Gatewayをどうするか，アドレスはどうするか，などということも重要なのですが，最も重要なことはコミュニティを作ることなのです．ただし，こういったコミュニティは，お偉いさんが集まって「ネットワーク何たら委員会」なんかを作っても何の役にも立たないのです．そもそも，こうした技術的なことを知らない人たちが決められることは，余計な「規則」が関の山でしょう．ともかく，実際に管理を行なうメンバーが集まった「草の根コミュニティ」を作ることが重要です．ただし，予算とかはそうした権限を持った人が必要ですから，「草の根コミュニティ」が人選して「ネットワーク管理委員会」なるものを作っておくことも忘れてはなりません．

「草の根コミュニティ」では，まずメンバーの技術力を知っておくことが重要です．ネットワークでは，1カ所でもドジをするとシステム全体が破綻をきたすことがあります．ですから，必要に応じてサポートを行なう体制も確立しておかなければなりません．もちろん，自分がサポートを受ける側だったりすることもありますよネ．そのときは，ただやってもらうのではなく，相手から知識を吸収するつもりで努力してください．最初は大変ですが，みんなの技術力がそろってくれば，あとはスムーズに作業が進むようになると思います．

こうして，「草の根コミュニティ」ができると，今度はコミュニティ間の連絡を密にすることが重要になってきます．計算機の運用スケジュール（停電やメンテ

ナンスで計算機が停止するといった情報)とか，設定変更の連絡，トラブル情報などは，速やかに伝達されるメカニズムを用意しておくことが望ましいでしょう(僕も，こうした連絡をよく忘れて失敗をやらかしていますが……).こうしたときに活躍するのがメーリングリストと呼ばれるものです.

たとえば，staffs@chofu-u.ac.jpというアドレスにメールを送ると，調布大学のネットワーク管理者全員に同じ内容の電子メールが届くという設定ができます.これをメーリングリストと呼びます．管理に関する情報はこうしたメーリングリストに送るという約束を作っておけば，あるところが古い設定のまま使っていて，うまくいかないなどというトラブルは減るはずです.

★

この連載で何度も話をしたように，Unixは利用者が作り上げてきた環境ですから，利用者の声も重要です．こうした利用者の声を受け取るための手段がローカルニュースグループです.

JUNETなどでは，BNewsと呼ばれる電子掲示板システムが用いられています.このシステムでは，話題に応じてニュースグループというものを作り，みんなで議論を行なうようになっています．このニュースグループは，世界中に配られるもの，日本国内だけ，組織内だけといった配る範囲の指定ができ，そのうち組織内（たとえば，大学内部）だけで用いられるニュースグループのことを，ローカルニュースグループと呼びます.

活発な組織では，ローカルニュースグループとして，さまざまなグループを用意して多くの議論が行なわれているようですが，最低限「管理者からのお知らせ」に使えるニュースグループを用意するようにしましょう.

こうしたニュースグループには，計算機運用スケジュールや設定の変更のお知らせだけでなく，新しく用意されたコマンドのお知らせなども登場するはずです.逆に，利用者からこういうフリーソフトウェアがあるようだから，インストールしてくれなんて「つきあげ」がきたりするかもしれません．このような，システムをうまく利用しないと,「ディスクの容量が足りなくなったので，よく調べてみたら，みんなが同じフリーソフトウェアを自分のディレクトリに抱え込んでいた」なんてことになってしまいます.

★

ところで，ネットワークの中にはファイルサーバやメールサーバといったみんなが共有して利用するマシンが存在します．こうしたマシンの管理は，複数の人によって行なわれますので，トラブルが発生しやすくなります．で，どうするかというと，前述のメーリングリストも重要なのですが，ノートを1冊用意するといいようです．このノートに，トラブルシュートや設定の変更を書き留めておくのです．これは，計算機上のファイルではなく紙のノートであるというところが，実は重要なのです.

こうしたノートは，トラブルが発生すると活躍するのですが，下手をすると計算機がうんともすんとも言わないなんて状態から復旧しなければならないこともあります．こんなときに，計算機の中のファイルに情報があっても何の役にも立ちません．ですから，紙のノートがいいのです.

このノートに蓄えられた情報は，トラブルシュートのKnow Howの蓄積にもなるので，あとで同様のトラブルが発生したときに復旧がしやすくなってきます(関係ないけど，1年ぐらい経ってからこのノートを読むとけっこう笑える事故がよくあります).

★

初期の立ち上げのときはともかく，定常的にネットワークが運用できるようになってくると，管理作業はトラブルとの闘いになってきます．ある程度，安定してシステムが動くようになると油断しがちですが，トラブルはそういう隙をついてきます．これまでに説明したような作業（メーリングリスト，ニュース，ノート）やバックアップ作業は，いつでも忘れずにやっておくように心掛けるべきでしょう（といいながら，僕自身さぼっていたりするが……．おかげで，先日地震で中核マシンのディスクがクラッシュしてえらい目にあってしまった).

★

まあともかく，多くの場合，本職以外にしかたなくシステムの管理を行なっている人が多いことでしょう(昔々，Super Users' AssociatesなんていうUnix管理者派遣会社を作ったら儲かるかなあなんて話を冗談でしていましたが，アメリカではこうしたビジネスがあるようです).ですから，管理をしてくれている人々の手間を少しでも減らすため，みんなで助け合って上手に運用のできる，正しい「草の根コミュニティ」を作りたいところですね.

Fat Bits

第4回　地球まるごと486

山崎 俊一

日本電気が80486／25MHzを搭載したPC-H98モデル100を発表した．1120×750ドットの高解像表示とFPU内蔵で，GUIに強そう．もっともっとWindowsといいたいけれど，定価は215万円（100Mbytesディスク内蔵タイプ）．個人で買える値段ではない．SPARC Stationよりは安い，ということだろうか．

でも，希望はある．米国でも初期の486 PCは1万5000ドル以上だった．1年後の今，7000～8000ドルの機種はザラにある．その気になれば5000ドル程度でボードから組み立てることもできる．

25MHz486CPUは，33MHz386チップセット（CPU＋FPU＋キャッシュ制御）よりも安いし，安いメモリが使える．基板も簡潔になり量産効果が上がるはず．低価格高性能は，インテルの486基本戦略で，いずれ国内のパソコンにも反映するだろう．

25MHz486が33MHz386よりも速いという不思議

386/33MHzのモデル70に対し，486/25MHzのモデル100は平均1.5倍速だという．

386/33MHzマシンは，286/10MHzマシンの4～5倍速い．これはクロック速度と32bit化の反映と理解できる．が，486と386では，ともに32bit幅で命令もメモリ管理も共通だ．つまりクロック速度と性能の関係が逆転している．浮動小数点演算を除外しても486が速い．なぜだろう．

最新の486は386より速くて当たり前，かもしれない．だけど，当たり前を疑うことから進歩が生まれる．そこをFatBitするのがこのページだ（と編集のX氏がけしかける）．

DRAM速度限界

少し話が細かくなる．CPUはクロック単位で動く．脈拍のようなものだ．メモリへのアクセスはアドレスを指定し，読み書きを行なう．それぞれ1クロック，計2クロックで1動作になる（これは386も486も同じ）．

CPUからアクセスされるメモリも，CPUと同じ速度以上でアクセス可能でなければならない．それ以下ではCPUが待ち状態（WS：Wait State）になる．が，DRAMの速度は限られていて，上限は60ns（ナノ秒），16MHz程度だ[*1]．

そこで，クロックが16MHzを超える場合，CPUと主メモリの間に高速なSRAMを置き，これをキャッシュメモリとする．CPUはキャッシュの範囲内でノーウェイト動作が可能になる．その効果は非常に大きい．でも，キャッシュの内容は主メモリから読み込まれ，主メモリに書き戻される．その度に起こる待ち状態は解決されない．

486バースト転送モード

このDRAMの限界に対して解決策はあるのか．ここで386と486の差がひらく．その違いは，486の内部キャッシュとバースト転送モードにある．

誇張していうと，486はバイトやワード単位の主メモリの読み書きを行なわない．常に4ワード連続で一括転送する．これをバースト転送と呼ぶ[*2]．

そんな馬鹿な．386でも486でもバイト，ワードはプログラムの基本単位だ．と思われるかもしれない．もちろん，486内部の演算ユニットと内部キャッシュの間は通常のやり取りをしている．しかし，キャッシュと外部メモリのやり取りでは，バースト転送が使われる（図1）．これはハードウェア機能であり，OSやソフトウェアは関与しない．486では，何もいわなくても自動的にこのモードが起動されるようになっている（具体的には/BRDYピ

＊1　概算で1/16MHz×2＝125ns．DRAMのアクセス速度でいうと65ns程度．
＊2　ハードウェアがバースト転送を実装した場合の話．I/O命令を除く．

ンがアサートされる).

ページモードRAM

バーストモードでは，連続するメモリ領域が続けて転送される．つまりアドレス指定は最初の1回だけ行ない(1クロック)，あとは順次読み出し動作だけ続ければよい（各1クロック）．

486のキャッシュは4ワード単位だから，従来の読み書きでは4×2=8クロックになる．しかしバーストモードなら5クロック．36.5%速い（**図2**）．

それだけではない．486の主メモリは，連続アクセスだけ考えればよい．するとページモードDRAMが使える*3．多少制約はあるが，70nsページモードDRAMのインターリーブ構成で，25MHz/486がノーウェイト動作可能といわれる．実際には，486と主メモリの間に，さらに外部キャッシュを置く機種が多い．

疑惑の“No Wait”

世の中，“ノーウェイト”とうたうパソコンは多い．が，その実現は本当に難しい*4．486はそこをうまく回避している．だから速い．この設計はスジが通っているし，納得できると思う．

ただし，内蔵キャッシュとバースト転送は（例によって）インテルのオリジナルではない．もとは汎用大型機の技術で，それがIC製造技術の進歩でワンチップCPUまで降りてきた．モトローラの68030も基本構成は共通だ．ページモードRAMはIBMがオリジナルだったと思う．

なお，米国の激速パソコンでは，メモリ構成やウェイト数，アクセス速度，キャッシュの方法，キャッシュコントローラ，その他もろもろのシステム情報が公表されているものが多い．そういう情報が豊富な機種が結局は普及する．モデル100はまだ見ていないけれど，そうなってほしいと期待する．

マルチメディア資本主義

この号が刊行されるころ，幕張メッセでマルチメディア国際会議が開催される．今年はW. Gates（Microsoft社）が来日する．何か爆弾を隠しているらしい．多分MacroMind社との連携作戦についてだろう．

次に11月14，15日，新宿でAVA'90という催しが開かれる．こちらは映像業界中心のマルチメディア・セミナー（展示見学無料）．その講演者を見ると，Robert Abel，S. Fisher(MIT出身，NASAで仮想環境ワークステーションとかやっていた人）などと並び，David Coleの名前がある．

どこかで聞いた名前だと思ったら，元Ashton-Tate社の会長．現在は，マルチメディア狙いのベンチャーキャピタリストだ．実はMacroMindも彼が出資している．

気が付いてみたら，マルチメディアも財テクの対象になっていた．マルチメディアもおもしろいけど，そういう業界人の生態ウォッチングはもっとおもしろそうだ．

(AVA'90事務局：☎03-797-1839)

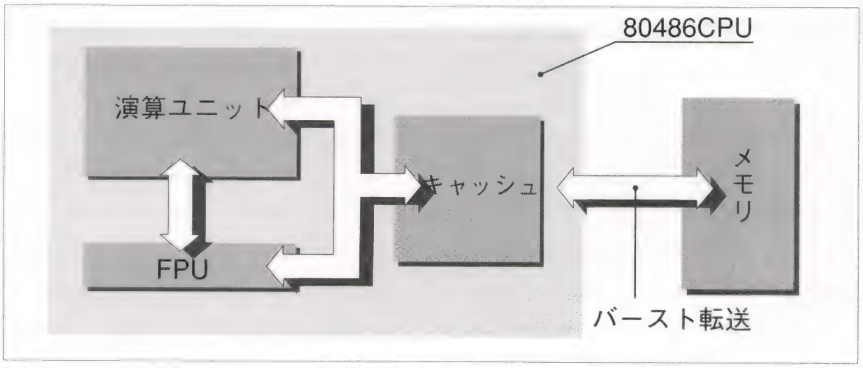


図1 486の構成．演算ユニットとキャッシュの間は通常動作．キャッシュと外部メモリの間でバースト転送を使う．

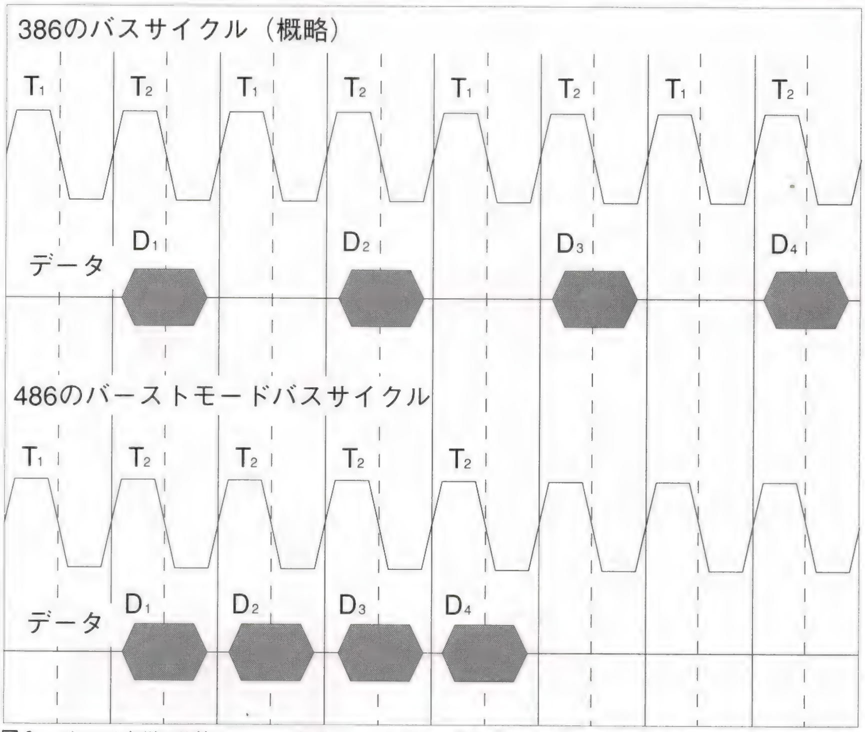
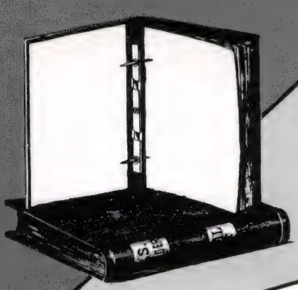


図2 バースト転送の比較．バースト転送では，アドレス指定（図中のT1サイクル）を1度すれば，その後は連続してデータをアクセスできる（T2サイクル）．

*3 一連のメモリ領域（＝ページ）を連続読み出しできるタイプのDRAM

*4 100%ノーウェイトを実現するには，全メモリを25nsSRAMにする，などという非現実的な話になる．たとえば主メモリをデュアルポートRAMとした機種（NCR製486マシン）もある．



BOOK SHELF

電子出版

白田耕作
株式会社日本電気文化センター

本書は，DTP（机上出版）やCTS（電算写植システム）などの制作工程の電子化ではなく，出版メディアの電子化という切り口で，“電子出版”をとらえている．これまでの紙に情報を印刷加工していた出版メディアに対して，電子出版メディアとは「音楽CD，その親類の光ディスク系のCD-ROM，さらにCD-ROMの親類で現在開発が進められている動画を扱うDV－I，普及を目指している再生装置込みの規格を持ち静止画を中心に扱うCD-Iなど」である．この本では，すでに1000点ほどが出版されているCD-ROMについて，規格や仕組みから応用例にいたるまでを多くのページを割いて解説し，その魅力を語っている．

なるほど書籍と張り合える

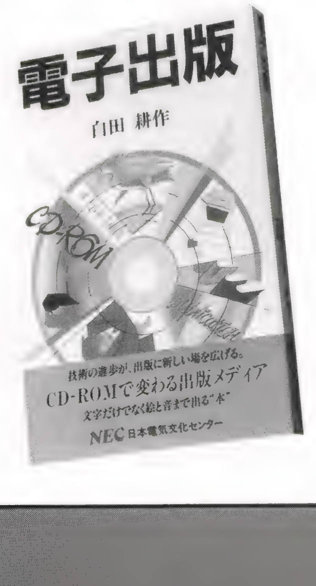
制作費は別だが，ディスク1枚の材料費は500円で，1Mbytes当たりにすると0.93円，と著者はいう．これなら，紙に印刷して製本するより安い．また，同じ市内に電話線を使って9600bpsで送信すると，1円当たり17Kbytes．時間はかかるが，CD-ROMなら900円の宅配便を使っても1円当たり600Kbytesが送れるという．こんな数字を挙げられると，確かに，電子メディアの中では，複製を大量に安く配るという，今までの“出版”と同じ特徴を持っていると再認識させられる．

出版の例としては，広辞苑，オックスフォード英語大辞典，朝日新聞記事データベース，現代用語の基礎知識が取り上げられ，検索例を示しながら特徴が解説されている．

広辞苑を例にとれば，“何とか族”とか“何とかメディア”とかいった後方一致の検索ができたり，検索した鳥に鳴き声が付いていたり，検索した色を実際に画面で見たりできるので，本のときより楽しい辞典になり，メディアとしての奥行きも出てくると私も思う．

私見・もっと身近にならないか

ただ，電子出版メディアが書籍のように図書館で閲覧できるかというと，「パソコンなどのハードや検索するためのプログラムも準備しなければならないため，国公立の図書館では取り扱いにくい」と国立国会図書館ではいう．



家庭でも，CD-ROMドライブを備えたパソコンがなければ使えないのでは，個人のメディアにはなりにくいと思う．

日本電気の話ではないが，最近の電子出版のトピックスは，今年7月にソニーから発売された，かばんに入れて持ち運びできるデータベースマシン“データ・ディスクマン”ではないだろうか．本体が5万8000円，8インチCD-ROMを使ったソフトは1万円以下のものが多く，個人でも手を出しやすそうだ．すでに2万台が売れているというが，著者には気になる製品かもしれない．

データベースを引いた体験から

CD-ROM版ではなく，この本でも少し触れられているオンライン検索だったが，新聞記事のデータベースは，私が記者をしていたときに大いに利用させてもらった経験がある．

自社で作って売り物にしているデータベースを社内で利用するというのは，奇妙に思われるかもしれないが，新聞記者といっても，いつ，何が載っていたかをすべて覚えていられるわけはない．同じ内容の記事がすでに載っていないかをチェックしたり，同じような事件の記録を検索したりするツールは「ぜひもの」である．

新聞の切り抜きも，テーマごとに分類

した封筒に入れて保存されているが，慣れてしまえば，キャビネットから探し出すよりデータベースを検索するほうが便利だった．

マルチメディアとハイパーメディア

本書の最後では，マルチメディア／ハイパーメディアは出版メディアかというテーマも論じている．

マルチメディア／ハイパーメディアは，最近コンピュータ関係の雑誌で特集が組まれたり，シンポジウムや展示会が開かれるなど，未来のメディアとして注目され始めている．この本から，それぞれのメディアを説明した部分を抜き出すと，マルチメディアは「文章，音声・音楽，ビデオ・動画，この中には，テレビ映像，ハイビジョン映像，高精度CG映像などがあり，さらに静止画など，個別の表現としてのメディアがある．これらを必要に応じて組み合わせ，総合的に活用するメディア」である．

ハイパーメディアは「さらに，このマルチメディアで表現される文書に対して情報間の関係をつけたり情報に広がりと深さを付与したり，情報に構造を持たせる　このような文書が持つ情報のマクロ的な関係づけを行なうと同時に，文章で使われている言葉のインデックスづけを行ない，文書に対する検索処理を効果的に行なえるようにミクロ的な関係づけを行なう．そのうえでこれらを対話的な操作により，使用者のいろいろの意味でのレベルに応じた能動的な情報獲得ができるように」したものである．

著者は，東京都港区の機械産業記念館で12月15日まで開かれている，“コンピュータと人間との新しいふれあい”をテーマにした展示会「HARP（ハイテク・アート・プランニング）」の展示作品の制作にかかわった．コンピュータグラフィックスを使ったNHKのテレビ番組『ヒューマン・ボディ』を“ハイパーメディア”に再構成したものである．

出版メディアとして成り立つためには，“安く，容易に，大勢の人の手に渡す”ことができなければならないと考える著者は，今回の制作体験を基に，出版メディアとこれらのメディアを比較して，「電子出版は書く言葉を中心にした出版活動の1つである．マルチメディアを活かすとしたら，書く言葉を中心に表現された著書に，若干の説明を補足するための図や，動きのある絵を使い，音楽や音声を使うことで，適切な表現となるならばそれを活用するということでよいと思う」と述べる．

動画を中心としてマルチメディア／ハイパーメディアをとらえ，万能のメディアのように考える最近の流れには慎重な態度をとる．

また，マルチメディアとハイパーメディアの関係について，「ハイパーメディアが一般性のある，広い範囲で活用できるメディアとして開発されてきてはいるが，マルチメディアを強調し過ぎて，制作の面で，膨大な時間と費用を必要とするメディアになることには注意が必要」という．

パソコンのマルチウィンドウに映し出されたフルカラーの映像を見て感激してしまう私からみれば，映像（動画）のないハイパーメディアなんて，と反論をしたい気もする．

「長い目で見れば，情報の収束現象，すなわち，放送，出版・印刷，コンピュータは1つにとけあっていくということが進むかもしれない」とはしながらも，「現在の技術をしっかりと役立てていき，それを発展させていくことが当面の課題である」と言われてしまうと，コンピュータがメディアの中心に位置するにはまだまだ時間がかかりそうで，寂しい気もするのである．

私見・近未来へ向けて

10月3日から8日まで，ドイツで開かれた，世界最大の出版関係の展示会「フランクフルト・ブックフェア」の今年のフォーカルテーマは“日本年”だった．日本の文化，伝統を伝えたいと，奈良時代の日本最古の印刷物，法隆寺の『百万塔陀羅尼（だらに）教経』から現代にいたる出版物が展示された．その中には，出版の近未来の可能性を示す出版物として百数十点のCD-ROMもあった．

日本は，米国についで，CD-ROMの発行点数が多い国だ．単に必要な情報が得られるだけではなく，関連する情報が柔軟に引き出せるという特徴をもっと発展させて，電子出版物が，我々の知的好奇心をかきたてるようなメディアに育ってほしいと思う．

ところでこの本の著者は

著者の白田さんは，本書と同じC&C文庫の『CGへの招待』を執筆したり，日本コンピュータ・グラフィックス協会（NICOGRAPH）の企画委員や「日経MIX」のCD-ROM会議の議長などを務めたりする“マルチタレント”（相互に関係し合うのでハイパータレントのほうが適切？）である．’87年には，朝日新聞・科学面の連載『変わるメディア』もある．

有料なら図書館がある

東京都新宿区には「情報図書館RUKIT」がある．入会金や会費などは必要だが，CD-ROMや内外のオンライン・データベースを利用することができる．

料金は，個人なら，入会金5000円，会費1000円／月，施設利用料は1データベース，1CD-ROMにつき500円，オンラインの場合は実費の検索料が必要，というふうになっている．

問い合わせは，☎03(266)9315まで．

小森真幸
朝日新聞制作局開発部
新聞制作にかかわる汎用大型計算機を中心としたシステムの開発に携わっている．近頃は，マルチメディアやハイパーメディアに興味がある．

Readers Board

このコーナーでは，本誌の内容に関して読者の方から寄せられたご意見・ご感想を掲載しています．今回は，9月号についていただいたお便りの中から紹介します．

From Readers

■WSが欲しい．でもviは嫌いだ！　Emacsも…もう一つ…うーん．X Windowもわけわからん！

大阪府　滝野圭三

この気持ち，よく分かります．「viのインサートモードのうっとおしさ」に頭をかきむしったり，「EmacsのM-x...」のコマンドのキーストロークに指がひきつるなど，結構四苦八苦していた記憶があります．今でもEmacs Lispには手が出ないので，.emacsの変更は人任せにしているのですが，この状況を情けないと思ったのか，某編集者がこのほどEmacs Lisp入門をスタートしました．これは「Lispを知らなくても大丈夫だから，猿でも分かりますよ」と，私を脅迫するので，これを機に，なんとかマスターしようと思っています（猿に負けたくない私）．

◈

■創刊号の編集後記で，あの元ログインの五十嵐さんの名前をみて，驚いていたら，今月号では，ご結婚されたとのこと．おめでとうございます．ログインのNo.14の墜落日誌の，あのなぞの花ずきんちゃんの雰囲気は，オタク五十嵐さんでは？　と思っていたので，やっぱりという感じでした．10月号の特集を今から楽しみにしています．

岡山県　吉川桂子

ログインを読んでいない方には分からないでしょうが，当編集部の五十嵐は“オタク五十嵐”として，かつてログインで名を売っていました．その当時，彼が担当していた「墜落日誌」というマンガの作者である寺島礼子さんに，ホニャララして，とうとうホニャ×してしまったのです．「編集者と女性マンガ家との結婚!!!」この世界ではよくある話ではありますが，彼が奥さんの話をするときの表情には，筆舌に尽くし難いものがあります．

◈

■オタク五十嵐や伊藤ガビンの名前が目に入った時は驚きました．ところで「新・日本サイバー」はいつ再開するのでしょうか．

山形県　渋谷　清

引き続き旧ログイン関係者のネタです．伊藤ガビンがＶＲの記事を書いたのは，あくまでも特別参加です．「新・日本サイバー」は，いつ再開するか？　これは誰にも分かりません．それにしても，スーパーアスキーの読者の中に，意外とログインの読者が多いのに驚いています．ところで，日本サイバーってどんな記事だっけ？

◈

■パソコンユーザーだってSuperASCIIを読むんです．Super Heavyな98の使い方など特集してください．

三重県　山田　浩司

今月号を見ていただければ分かるように，スーパーアスキーでは，パソコンの記事を扱わないわけではありません（今のところ少ないけど）．ちょうど，創刊からこれまでの時期に，パソコン関係のトピックが少なかったこともあり，２号目までは98関係の記事がなかったのですが，今後はいろいろな形でパソコンの記事も取り上げていきます．

Postscript

◆ASCII NET PCSにご加入の方はご存じと思いますが，月刊ASCII／ハッカーズ・コネクションというノートファイルに，スーパーアスキーの感想ノートを開いております．メニューからの番号選択でもよいのですが，MENU>>のプロンプトのところで，open salon.hackes，または，/3/9/5/1と入力しますと，ダイレクトにsalon.hackersにいけます．インデックスが出ますので，INDEX>>のプロンプトのところで読みたいベースノートの番号を入力してください．レスポンスの付け方など詳しいことはマニュアルなどをご参照になり，こちらへもご意見，ご感想をお寄せくだされば幸いです．

◆アンケートハガキによる読者プレゼントの締め切りは，p.173にあるように毎月10日（必着）です．つまり，毎月18日が発売日ですが（18日が日曜日の場合は17日になります），翌月の10日までに届いた分が，その号で紹介したプレゼント抽選の対象となります．また，p.175にあるように，読者プレゼントに当選しなかったハガキの中からも500円の図書券を抽選で50名様にお贈りしています．ぜひお早めにご返送ください．10日を過ぎて届くハガキもたくさんあります．競争率がかえって高くなってしまったではないか！　と，一部の方からは叱られそうですが，念のためお知らせまで．

お知らせ

スーパーアスキー編集部では，以下の要項で，アルバイト（編集サポートおよびプログラミングスタッフ）を募集しています．

■資格：18歳から25歳位までの学生の方で，週に２～３日以上当編集部で勤務できる方．

■待遇：時給700円以上，交通費支給．

■締切：11月末日．おって面接日などをお知らせ致します．

■応募方法：履歴書（面接日，連絡方法の希望があればそれも明記）および「私の欲しいコンピュータ」(600字程度．なるべくワープロ原稿で）を必ず封書にて，以下の宛先までお送りください．

■宛先：〒107-24　東京都港区南青山6-11-1
スリーエフ南青山ビル
(株)アスキー　人事部　「スーパーアスキー編集部アルバイト」係

Presents.

SuperASCII 読者プレゼント

今月の読者プレゼントは次のとおりです.

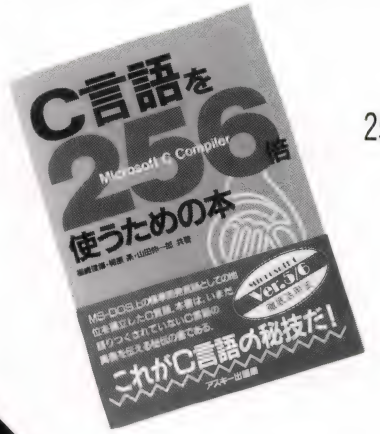
1

腕時計

3名

日本サン・マイクロシステムズ(株)より，同社のオリジナル腕時計を.

2

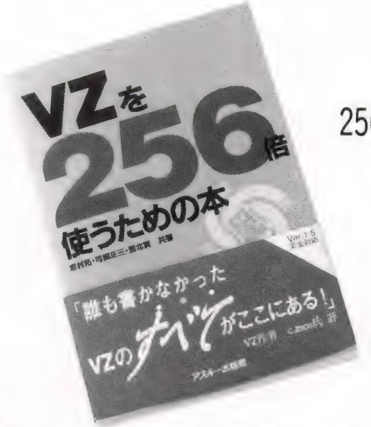


C言語を256倍使うための本

5名

C言語をもっともっと使いこなしたいあなたに，ワンステップ上のプログラミングをお約束する新刊書.

3



VZを256倍使うための本

5名

VZ使いこなしのノウハウをぎっしり凝縮．大小マクロ，ゲーム14本を全公開したVZユーザーの必読書.

4

プログラミングOS/2 PMプログラミング編III, IV

5名

OS/2の標準グラフィック環境であるプレゼンテーションマネージャーのプログラミングノウハウ満載書.

応募方法

読者プレゼントをご希望の方は，Super EXPRESS (p.17) の前にとじ込んであるアンケート・ハガキでお申し込みください．裏面に記入欄があります．希望番号と商品名をお書きください．応募者多数の場合は抽選になります.

なお，本誌記事に関するアンケートには必ずお答えください．プレゼント申し込みの締め切りは11月10日(必着)です．当選者の発表は発送をもって代えさせていただきます.

最新ソフトウェアバージョンリスト

Software Version List

システムソフトウェア，ソフトウェアツールなどの最新動向が分かる，プログラマのためのバージョンリスト

	名称	バージョン	ベンダー	対応機種	リリース	備考
	SunOS	4.1	サン	SPARC Sun-3, Sun-4	1990年4月	SunOS 4.0とバイナリ互換 System V Release 4とソース互換
	SunView	1.8	サン	SPARC Sun-3, Sun-4	1990年4月	
	OPEN LOOK	2.0	サン	SPARC Sun-386i Sun-3, Sun-4	1990年7月	
	HP-UX	7.0	YHP	HP9000 S300, S800	1989年12月	
		7.03	YHP	HP Apollo 9000 S400	1990年7月	
○	DOMAIN-OS	SR10.3	YHP	Domain HP Apollo 9000 S400	1990年9月	
	motif	1.0	YHP	HP9000 S300, S800 HP Apollo 9000 S400	1989年8月	
				Domain	1990年1月	
○	UniOS-U	1.4	オムロン	LUNA LUNA-GW LUNA-FS	1990年9月	統合化UNIX
	UniOS-B	1.5	オムロン	LUNA LUNA-GW デスクトップ	1990年2月	UNIX 4.3BSD
	UniOS-Σ	1.3	オムロン	LUNA-Σ	1989年11月	ΣOS（V0R1）準拠
	UniOS-Mach	1.0	オムロン	LUNA-88K	1990年8月	Mach Ver.2.5準拠
	MS-DOS	3.3C	日本電気	PC-9800 (98LT, H98を除く)	1990年7月	
		3.3C	日本電気	PC-H98	1989年7月	
		3.1	日本電気	PC-98LT	1986年10月	
		J4.0	日本IBM	IBM PS55	1989年5月	日本語版，英語版は1988年8月発売
		4.0	マイクロソフト	IBM PC, PS/2	1988年8月	
	MS-WINDOWS	2.11	日本電気	PC-9800 (98LT, H98を除く)	1989年10月	日本語MS-DOS Ver.3.1以上が必要
		2.11	日本電気	PC-H98	1990年1月	PC-H98専用の日本語MS-DOSが必要
		2.1	日本IBM	IBM PS55	1990年2月	日本語版
		3.0	マイクロソフト	IBM PC, PS/2	1990年5月	
	MS-OS/2	1.1A	日本電気	PC-9800 (98LT, H98を除く)	1990年7月	メモリは4.5Mbytes，要HDD（20Mbytes以上）
		1.1A	日本電気	PC-H98	1990年7月	PC-H98専用の日本語MS-DOSが必要
		J1.2	日本IBM	IBM PS55	1990年3月	日本語版，英語版は1989年5月発売
		1.1	マイクロソフト	IBM PC, PS/2	1988年8月	
	MS-C	5.1	マイクロソフト	PC-9800	1988年8月	Ver.6.0は秋に出荷の予定
		6.0	マイクロソフト	IBM PC, PS/2	1990年4月	
	MASM	5.1	マイクロソフト	PC-9800	1988年9月	
		5.1	マイクロソフト	IBM PC, PS/2	1988年7月	
	Turbo-C	2.0	ボーランド	PC-9800	1988年12月	MS-DOS 2.1以上
		2.0	ボーランド	IBM PC, PS/2	1988年8月	PC-DOS 2.0以上
		2.0	ボーランド	各種AXマシン	1989年7月	MS-DOS 2.1以上
		2.0	ボーランド	FM-R Panacom M	1989年11月	MS-DOS 2.0以上
	Turbo-C++	1.0	ボーランド	IBM PC, PS/2	1990年5月	PC-DOS 2.0以上
	Turbo-Pascal	5.5	ボーランド	PC-9800	1990年1月	MS-DOS 2.1以上
		5.5	ボーランド	IBM PC, PS/2	1989年5月	PC-DOS 2.0以上
		5.0	ボーランド	FM-R Panacom M	1989年11月	MS-DOS 2.1以上
	Turbo-Debugger &Tools	2.0	ボーランド	PC-9800	1990年7月	MS-DOS 2.1以上
		2.0	ボーランド	IBM PC, PS/2	1990年5月	PC-DOS 2.0以上
	gawk	2.11.1	GNU	UNIX, MS-DOS	1989年11月	日本語対応のjgawkは，2.11.1をベースにしてあり，バージョンは+2.0
	bison	1.03	GNU	UNIX, MS-DOS	1989年8月	
	emacs	18.55	GNU	UNIX	1989年8月	
	gcc	1.37.1	GNU	UNIX	1990年2月	
	g++	1.37.1	GNU	UNIX	1990年3月	
	make	3.58	GNU	UNIX, MS-DOS	1990年2月	
	scheme	7.0	GNU	UNIX	1990年6月	
	smalltalk	1.1	GNU	UNIX	1990年5月	
○	file utilites	1.4	GNU	UNIX	1990年9月	内容は，cat, chmod, cmp, cp, cut, dd, dir, du, head, install, ln, ls, mkdir, mkfifo, mv, paste, rm, rmdir, tac, tail, touch, vdir

表の左端はチェック欄で，最近大きな変化（メジャー番号の変更）があった場合に◎が，小さな変化（マイナー番号の変更）があった場合には○がついている．対応機種については，原則としてシリーズ名で表示しているが，特定の機種のみに対応しているものについては具体的に表示している．リリース欄にある日付は，掲載した最新バージョンの発売（出荷）時期である．データは，9月上旬の調査によるものであり，それ以降にバージョンアップが行なわれた可能性がある．

アンケート・ハガキの記入方法

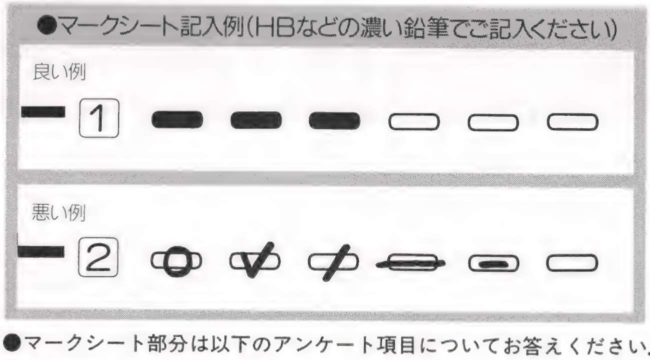
アンケート・ハガキは，本誌Super EXPRESS(p.17)の前にとじ込んであります．アンケート・ハガキの表面の空白部分には，スーパーアスキーへのご意見・ご感想などをお書きください．イラストなども歓迎いたします．書いていただいた文章などは本誌Readers Board欄に掲載させていただくことがあります．

裏面のマークシート欄は，以下に挙げる質問について，該当する番号の枠内を記入例のようにHBなどの濃い鉛筆でマークしてください．ハガキには，質問事項が書かれていませんので，番号をよくお確かめのうえ，ご記入くださるようにお願いします．マークシート以外の部分，およびハガキの表面は鉛筆以外のペンで記入していただいてかまいません．

皆様からお送りいただいたアンケート・ハガキは，スーパーアスキーの内容をより充実したものにするための参考にさせていただきます．ぜひご協力ください．「読者プレゼント希望」の欄には，p.173に掲載したプレゼントの中から希望するプレゼントの番号および商品名をお書きください(希望する物がない場合は空白にしておいてください)．

毎月10日までに到着したハガキの中より，抽選で50名の方に図書券500円分をお贈りいたします(ただし，プレゼントの抽選を優先して行ないます)．なお，当選者の発表は発送をもって代えさせていただきます．また，Readers Board欄に採用させていただいた方には，図書券やオリジナル・グッズなどをさしあげます．

ハガキは切り取り線からきれいに切り離し，折り曲げたり汚したりしないようにご注意ください．



●マークシート部分は以下のアンケート項目についてお答えください．

1次の中から面白かった記事の番号を挙げてください（5つまで）．

①表紙
②Super EXPRESS
③イベントカレンダー
④特集・RISCはなぜ速いのか／RISCとは何か
⑤　〃　／RISCの実現
⑥INSIDE SOFTWARE　Windowsプログラミングとは何か
⑦TEST LAB./PC-9800シリーズの新製品
⑧　〃　/Hyper Card 2.0
⑨　〃　/RISC NEWS
⑩Hardware=Software＋Hardware
⑪特別企画　PostScript Level 2の全貌
⑫最先端システム探検隊
⑬Astro Guide '90
⑭プロフェッショナルワークステーションとしてのUNIX入門
⑮OS/2アーキテクチャ入門
⑯プログラミング・フォーラム／386 de MS-DOS
⑰　〃　／Emacs Lisp入門
⑱　〃　／ウィンドウシステムのプログラミングパラダイム
⑲EPnews
⑳MIPS／パソコン1024夜
㉑　〃　／プログラマのための数学的世界観入門
㉒　〃　／ウサギと月と，そして太陽と
㉓　〃　／Fat Bits
㉔　〃　／Book Shelf
㉕　〃　／Readers Board
㉖　〃　／最新ソフトウェアバージョンリスト

2つまらなかった記事の番号を挙げてください（5つまで）．

上記の項目の中からお選びください．

3本誌以外にご購読のパソコン雑誌は何ですか(いくつでも)．

①日経パソコン②日経バイト③日経コンピュータ④日経エレクトロニクス⑤The BASIC⑥パソコン・マガジン⑦THE COMPUTER⑧インターフェース⑨トランジスタ技術⑩bit⑪月刊ASCII⑫UNIX MAGAZINE⑬NETWORKER⑭その他

4本誌の購読についてお聞かせください．

①ほとんど毎号②ときどき③初めて

5コンピュータの使用状況をお知らせください（いくつでも）．

①自分で所有しているコンピュータを使用
②家族や友人のコンピュータを借りて使用
③勤務先や学校のコンピュータを使用
④まったく使っていない

6コンピュータを使用する目的は何ですか（いくつでも）．

①仕事②学業③趣味・遊び

7コンピュータは主にどのようなことにお使いですか（いくつでも）．

①ゲーム②ワープロ③データベース④雑誌などのプログラムリストの入力⑤自分でプログラムを作成⑥パソコン通信⑦コンピュータ・ミュージック⑧コンピュータ・グラフィックス⑨表計算・作表⑩技術計算⑪コンピュータによる教育・学習⑫パソコン自体の学習⑬その他

8次に購入したいと思っているものは何ですか（1つだけ）．

①コンピュータ本体②ハードディスク③拡張メモリ④プリンタ⑤スキャナ⑥モデムおよび通信ソフト⑦CG関連のソフトまたはハード⑧コンピュータ・ミュージック関連のソフトまたはハード⑨表計算・データベースソフト⑩ワープロ・エディタ⑪ゲームソフト⑫CやBASICなどの言語処理系⑬ユーティリティソフト⑭その他

9年齢

①10歳未満②10～14歳③15～19歳④20～24歳⑤25～29歳⑥30～34歳⑦35～39歳⑧40～44歳⑨45～49歳⑩50歳以上

10性別

①男性②女性

11職種（1つだけ）

①小学生②中学生③高校生④短大・専門学校・予備校生⑤大学・大学院生⑥技能・労務職⑦営業・販売職⑧専門・技術職⑨事務職⑩管理職⑪自営業⑫その他⑬無職

Editor's Note

パソコンとワークステーション

▽創刊以来，多くのご意見が本誌に寄せられている．その中で特に目立つものの1つに，「この雑誌は，パソコン雑誌なのか，UNIXワークステーション雑誌なのかはっきりしない．この際，ワークステーション専門誌にすべきではないか」というものがある．確かに，テーマを限定すればするほどターゲットは明確になる．しかし，我々が興味の対象とする「個人で利用する強力なコンピュータ」という範囲には，高機能化するパソコンと，低価格化するワークステーションの2つが存在する．その一方を専門に語ることと，両者を1つの視点から語ることとの差は大きい．

▽低価格ワークステーション動向の中で，今月注目に値するのはNeXTの新機種だろう．あのスティーブ・ジョブズが1988年に発表したNeXTの後継機である．NeXTは，当初からDisplayPostScriptの採用，MachおよびNextStepの搭載など話題の多い低価格ワークステーションだった．今回はCPUを68030の16MHzから68040の25MHzへと強化し，しかも低価格版は約5000ドルと，さらに価格を下げた．先のモデルでは標準搭載だったMOをはずすなど仕様の違いはあるが，コンセプトのみならずリアリティも強化され，今後もNeXTから目を離せそうにない．

▽一方で，今月号のTEST LAB．で紹介したPC-H98model100は，日本を代表するMS-DOSパソコンの上位機種である．80486の25MHzを搭載したこのマシンで何をするのか？ネットワークのサーバに使われることもあるだろうが，「単に，従来のソフトを利用するだけなら，ここまでの仕様はいらない」という意見も，価格さえ考えなければ，そのまま鵜呑みにはできない．

▽パソコンであれワークステーションであれ，このレベルの競争は激しい．自分にとって最適なマシンはどれか？　本誌で見極めていただければ幸いである．（土田米一）

編集後記

■今年の秋は，ホテルに泊まって，ディズニーランドに行って，おいしいディナーを食べて，などという非常に小市民的でしかも少女趣味的な夢のような予定を立てていたら，本当に夢になってしまった．今度立てる計画は，もっと非現実的なものにしたほうがいいのだろうか．（土屋）

■高原の道路には野菜の無人販売所がおよそ100メートルおきに立っている．ところが，商品の内容や価格には，販売所により微妙な違いがある．このような環境で，必要な晩ごはんの材料を低コスト，少ない停止回数でそろえるための計画を，分散スーパーマーケット問題と名付けたい．（五十嵐）

■Macintosh II fxが来た．ワークステーションよりも高価格なこの超高級マシン，当たり前のことであるが，従来機種と使い勝手は一緒．マシンのパワーを存分に発揮させるソフトを使っていないせいもあるが，なんだか損した気分ではある．さて，A/UX2.0でも買ってくるとしよう．（JIN）

■わが家は，トイレ，洗面所，風呂場の電灯用スイッチが3つ並んでいる．かれこれ引っ越して2カ月になるが，いまだにこの配列を覚えられない．そこでとりあえず全部ONにして，試行錯誤の末，必要のないスイッチをOFFするという作業を行なっている．情けない話だ．（小川）

■昨今は生活のリズムも回復しつつあり，おかげで何となく退屈な日々である．ま，何かと騒ぎを起こしてくれる楽しい友人もいるのだが，それだけでは少々物足りない．かくして，サンデー暴走族はまたしてもCA18の歌声とともに，中央高速を疾走するのであった．（ToSHI）

■厳しく長い暑さもやっと終わったようで，これからはラーメンが恋しくなる季節だ．東京中野駅の近くに知る人ぞ知る和風ラーメンの店がある．ここの店の名前をとった山の幸たっぷりの“おりづるらーめん”や，ひりひり辛い“たんづる”の人気は格別である．（沼尾）

■出がけにポストをのぞくと，SAS（有名なダイビング機材メーカー）から封筒が届いていた．愛用のBCD（ダイビングのとき着るチョッキみたいなもの）が欠陥で，回収するという知らせだった．これから海の透明度も上がり最高のシーズンになるのになんということなのだろう．（こばやし）

■故あってこのところ富士山麓のキャンプ場に出入りしている．原生林を切り開いてキャンプサイトを造り出し，伐採した木ではカヤ葺きのあずまやを建てるなど，開拓の過程を見られるのが面白いからだ．今までは見ているだけだったが，そのうち手伝いに加わろうと思う．（小田切）

■友人から電話があった．北海道旅行に行くという．私の出身地だ．彼女は北海道に憧れていて，私の帰省ついでに2度連れて行った．彼女の夢みる大草原や大牧場は，残念ながら私の街にはない．“サイロは？”“キタキツネはどこ？”と責めていた彼女の顔を思い出した．（面野）

■この世に生を授けられたら，個と種の保存のためにとりあえず無我夢中で生きるのが定めと考えていたが，この歳になってようやく子供ができた．自分ではどちらでもよかったのだが「どちらですか？」との問いに「男」と答えると，「よかったですね」と言う人が多いのは1つの発見．（土田）

月刊
スーパーアスキー12月号は11月17日（土曜日）発売！

特集 ユーザーインターフェイスはグラフィカルであるべきか？
Gnu Emacs vs. X Window

コンピュータを制御して仕事をさせるという立場からは，Gnu Emacsが持っているような2キーか3キーですべての指示ができるインターフェイスが便利である．しかし，何をしたらいいのかコンピュータから提示してもらうためには，GUI（Graphical User Interface）が有効である．いずれにしろコンピュータの性能が上がり，ユーザーのコンピュータへの要求事項がハードになるにつれ，ユーザーとコンピュータのインタラクションは密接になっていくに違いない．

●発行人　塚本慶一郎　●編集長　土田米一　●AD　（株）ポパイ　面野かおる　●副編集長　土屋信明　●編集　五十嵐久和／永瀬　仁／小川誉久／渡邊利和／沼尾禮子／小林章彦／水沼裕子　●制作　（株）ポパイ　江面信幸／高木由美／吉積里枝／山口美和子／高岡真理子　●表紙デザイン　村西順一　●表紙撮影　岡村義和　●進行　小田切秀人／毛利りえ　●撮影　小泉達芳／（株）ボイスカンパニー　●校正　秋山真実　●協力　上野和明／遠藤孝信／谷口　崇／柳沢俊行／菅野康一／北原直樹／風穴　江　●広告　佐藤敏明／秋山伴道　●営業　渡辺昌之／鷲山直己　●資材管理　勝又俊永　●印刷　凸版印刷（株）
●各記事ごとの著作者名は該当する記事中に表示しました．

編集・著作　**株式会社アスキー**

〒107-24　東京都港区南青山6-11-1　スリーエフ南青山ビル
Tel（03）486-7111（大代表）
情報（内容のお問い合わせ）Tel（03）498-0299（ダイヤルイン）
雑誌営業部Tel（03）486-1977（ダイヤルイン）　1719011

Brief 日本語

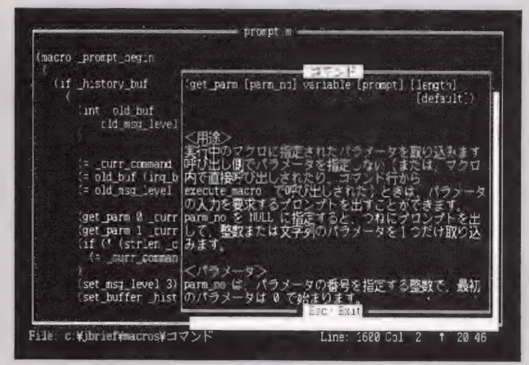
■Full Screen Editor

IBM PCでトップセールスに輝く
プロフェッショナル エディター「BRIEF」

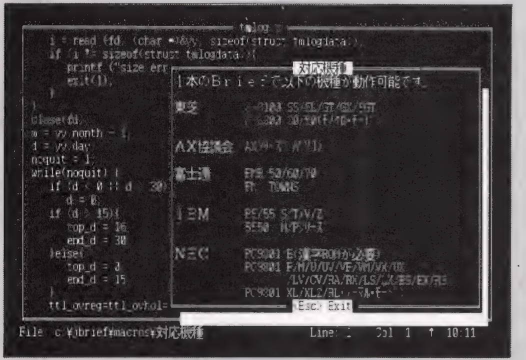
ブリーフの高機能 その①

■「メモ」「書式」「スケジュール」etc……

編集中も表示可能、マクロの応用機能。



メモ画面



対応機種一覧

マクロは「工夫」を実現します!

たとえば、マクロでオーバーラップ・ウインドウを表示することができます。表示だけでなくウインドウの中に書き込むこともできます。もちろんアンドウも使用できます。これを使えばいろいろなことができそうです。

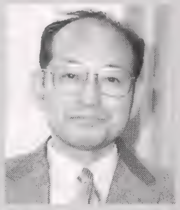
編集中に電話番号やちょっとしたメモが必要になったりします。そのたびに住所録やメモ帳を取り出すのは面倒なものです。そんなときはオーバーラップ・ウインドに表示させましょう。メモを見たり、書き込んだり、机の上を探し回る無駄な時間は最小限にしたいものです。

使わない方にも便利なマクロです!

「自分で作らないからマクロ機能はいらない!」と決めつけないでください。公開されたマクロを手にいれれば新しい機能を使うことができるのです。作らなくても機能を増やせることになります。このため弊社では、公開マクロのユーザー相互利用を推進しております。Wordstar用のキー・アサインや行のソート処理など魅力的なマクロを公開しています。Briefマクロプログラミング・コンテスト(8月末締切)の応募マクロも公開する予定です。公開マクロについてはBriefユーザー・サポート係にお問い合わせください。

USER'S VOICE

Briefとは長い付き合いになると思う。



私どもの実験は非常に厳しい条件で行うためデータの記録には迅速かつ正確さが要求されます。記録に漏れがあればデータの分析ができなくなり、全ての努力が無駄になります。このためマクロを利用し、記録項目をガイドに従って入力させることにしました。また、入力ミスにはアンドウで対処することで、迅速かつ正確な入力が実現できました。
日本語Briefとこれから先は長い付き合いとなると思います。

東京理科大学 工学部経営工学科
富澤儀一助教授

●革命的マルチマシン・エディター

東芝・NEC・AX協議会・IBM・富士通の5機種のパソコンで動作します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元に戻すことができますので、やり直しの面倒を気にせず、安心して編集できます。

●55ウィンドウ、無限のファイル数

ウィンドウは縦横自由に55まで開くことができます。また同時に編集できるファイルの数や大きさには制限がありません。

●プログラミング

◆自動コンパイル、エラージャンプ◆スワップアウト◆自動インデント、自動テンプレート

●安全編集

Briefは、一定時間キーを操作しない場合、その時点の編集内容をファイルに自動保存しますので、編集中に電源が切れても安心です。

Briefマクロプログラミング・コンテスト終了のお知らせ

Briefマクロプログラミング・コンテストは平成2年8月末日をもって終了いたしました。審査結果は、12月発表予定です。ご期待ください。

オピニオンリーダー誌のこの評価

- プログラミング向きのエディターを探している人は、もう探す必要はない。Jerry Pournelle(BYTE)
- Briefはプログラマーに最適なだけでなく創造力をかきたてる何物かがある。Computer World
- ほとんど完ぺきにカスタマイズできるだけの拡張性がある。Computer language
- まるで夢のようなアンドゥだ おもわず間違うのを期待してしまいそうだ。The C Users Journal

■主な機能

◆バッファ書出 切換 削除 リスト表示◆ファイル名の指定 表示 変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク/始・終点の入換◆ブロックのカット/コピー/ペースト 印刷 保存 左右へのシフト◆対応する括弧へ移動◆C言語のコメント化 非コメント化◆DOSコマンドの実行◆キー操作記憶/実行◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換 その他

定価 **48,000円**(消費税は含まず)

■全機種対応パッケージ

5' 2DD/3.5' 2DD一括提供(MS-DOS3.1以降対応)
上記以外のメディアは注文販売となります。
英語用DOSでは使用出来ません。

日本システム株式会社

●お問い合せは 東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL.0425-23-4469 FAX.0425-27-6127

FSI Fuji Software Inc.

FSI Original Products

富士ソフトウエアではMS OS/2プレゼンテーションマネジャ及びUNIX用アプリケーション、フォントの開発・販売を致しております。販売形態はOEM(ソース及びオブジェクト)、パッケージ販売などご要望に応じます。また以下のソフトを基にご希望のシステムの開発も承ります。

OS/2

PM対応日本語FEP Felix/2(仮称)

- 標準日本語環境であるKKC(Kana To Kanji Conversion Module)に準拠、本体はDLL(Dynamic Link Library)として提供します。
- ワープロ、データベースなどのそれぞれのウインドウに対して、きめ細かな環境設定が可能です。
- プレゼンテーションマネジャの機能を生かしたビジュアルなインタフェースを採用し、キーのカスタマイズ、サイクリック変換などのユーザーフレンドリーな操作性を実現します。

PM対応 Desktop Accessory Pocket Works(仮称)

- Pocket Worksは文房具感覚で使える、便利なアクセサリ集です。

【ワープロ】簡単な1ページ程度の書類なら、文章、表を同時に作成できます。
【電卓】演算は10進、16進ともに16桁まで演算可能です。ロールペーパー機能で計算過程が一目でわかり、演算結果のカット&ペーストも可能です。
【時計】アナログとデジタルで二つの時刻を設定可能です。またアラーム機能も装備しています。
【タイマー】アプリケーション実行中でも、アラーム音または画面フラッシュでタイムアップを知らせます。
【メモ帳】思いついたときにいつでもメモでき、10ページ(12行24桁/ページ)まで入力可能です。
【電話帳】氏名、住所、電話番号などを200件登録できます。ソート、検索も可能です。

GPIBデバイスドライバ

- MS OS/2用のバイモーダルなデバイスドライバです。非同期割り込み及び非同期入出力をサポートしていますので、MS-DOS環境では難しかったサービスリクエストの受け付けからシリアルポール、データ転送に至る一連のプログラムをマルチタスク対応で書くことができます。また他のデバイスドライバと並行動作ができるように、割り込み処理ルーチンはコンパクトに作られています。OS/2のAPIにスタイルを合わせたDLLも併せて提供致します。

イベントレコーダー(仮称)

- イベントの監視などのシステム情報の収集を行い、システムの保守、監視を容易にします。

オートパイロット(仮称)

- PM上で行うオペレーションを記録し、記録データの一部を加工することにより、バッチ処理やオートデモンストレーションを実行します。

UNIX

かな漢字変換ライブラリ SACK

- 日本語処理機能をサブルーチンとして提供致します。C言語で記述していますので移植性に優れています。また端末データベースを使用しているので、特定の端末に依存しません。

辞書

- 5万語及び3万語辞書を用意しております。テキスト形式なので辞書のメンテナンス、他OSへのコンバートが容易に行えます。

毛筆行書体メタフォント

- UNIX用毛筆スケーラブルフォントです。必要な大きさのTEX用フォントを生成することができます。

FONT

明朝体(開発中)
ゴシック体(開発中)
毛筆行書体(JIS第1・第2水準)
毛筆楷書体(JIS第1・第2水準)
イラストフォント(300種)

- アウトライン、ビットマップ、各種データフォーマットに対応致します。

※MS-DOSは米国Microsoft社の登録商標です
※MS OS/2は米国Microsoft社の商標です
※UNIXは米国AT&Tが開発し、ライセンスしています

OS/2 and UNIX

データショウ'90
Felix/2, Pocket Worksを出展致します。
- 会期/10月22日(月)~25日(木)
- 会場/東京国際見本市会場(晴海)
- ブース/新館1階 N-64

●開発・販売
FSI 富士ソフトウエア株式会社
〒108 東京都港区三田3-4-3 三田第一長岡ビル
営業部 担当:荒川/椿
TEL03-455-7731 FAX03-453-4643

資料請求 No.178

NBCC Communications & Image Processing

IBM・PC/ATコンパチ
パーソナル・コンピュータ
シリーズ

いますぐWINDOWS 3.0!!

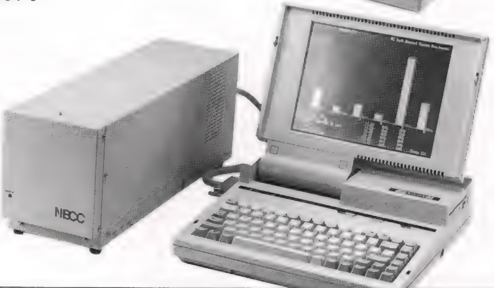
「高機能でしかもリーズナブルなAT・コンパチ・パソコンを」と言う要望にお応えして、NBCCパーソナルコンピュータシリーズのラインナップを充実させました。WINDOWS 3.0完全対応‼ コンパイルマシン、各種計測機のコントローラー、LAN構築の際のサーバーマシン及び端末としての利用を考えた386SX/16MHz採用高機能省スペース、ラップトップ、ノートブックパソコンを新発売。386エンハンスト・モード動作を考慮し拡張メモリをお求め安い価格設定に致しました。

新発売

NBCC 386AT

●高機能低価格を実現●VGAに対応、16階調表示を実現●CPUに80386SX/クロック16MHzを採用し各種業務処理に対応可能●メインメモリは、1MB標準装備、最大5MBに増設可能●40MBの高速・大容量ハードディスクを標準搭載●3.5インチ・フロッピーディスク内蔵●外部カラー・ディスプレイ(VGA対応)接続可能な出力コネクタ標準装備●東芝J-3100/T-3100互換、拡張スロット内蔵●1シリアル、2パラレル・ポートを標準装備●オプションの拡張ボックスを利用すれば5スロットまでXT/ATカード使用可能。

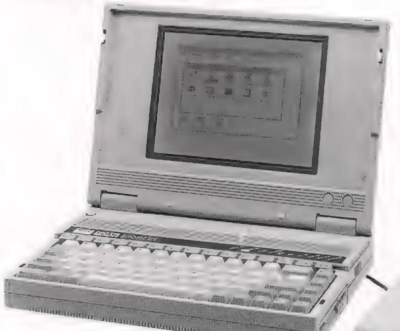
定価 390,000円



近日発売

NBCC NOTE BOOK 386/286

定価 380,000円/350,000円



●高機能低価格を実現●VGAに対応、16階調表示を実現●CPUに80386SX/クロック16MHz(80C286/12MHz)を採用し各種業務処理に対応可能●メインメモリは、1MB標準装備、最大で5MBに増設可能●20MBの高速・大容量ハードディスクを標準搭載●3.5インチ・フロッピーディスク内蔵●外部カラー・ディスプレイ(VGA対応)接続可能な出力コネクタ標準装備●ATコンパチ拡張インタフェイス装備●1シリアル、2パラレル・ポートを標準装備●オプションの拡張ボックスを利用すれば5スロットまでXT/ATカード使用可能。

NBCC 286LT

●お求め安い価格設定●VGAに対応、16階調表示、取り外し可能なディスプレイを標準装備、OHP-LCDモジュール(150,000円オプション)取り付け可能●CPUに80C286/クロック16MHzを採用し各種業務処理に対応可能●メインメモリは、1MB標準装備、最大で5MBに増設可能●40MBの高速・大容量ハードディスクを標準搭載●3.5インチ・フロッピーディスク内蔵●外部カラー・ディスプレイ(VGA対応)接続可能な出力コネクタ標準装備●東芝J-3100/T-3100互換、拡張スロット内蔵●1シリアル、2パラレル・ポートを標準装備●オプションの拡張ボックスを利用すれば5スロットまでXT/ATカード使用可能。

定価 360,000円

キャンペーン実施中‼ 平成2年10月15日〜平成3年1月15日(当日消印有効)

ご応募頂いた方より抽選で10名の方に386ATを1ヵ月間無料モニター、さらに上記外の方に抽選で2台のNBCC386ATをプレゼント

【応募方法】御住所・御氏名・年齢・勤務先名、御住所・下記アンケートをご記入のうえ左隅応募シールを官製葉書に貼付しNBCCパーソナル・コンピュータ・グループ キャンペーン係迄ご応募ください

【アンケート内容】1. 現在お使いのコンピュータは? 2. ATをどのような用途での使用を考えておられますか? 3. 使ってみたいATソフトは? 4. あなたがATコンパチ機の情報を得る方法は? 5. 御加入されているパソコン通信ネットは?

(本価格に消費税は含まれておりません) ※IBMはアイ・ビー・エム社の登録商標です。※MS-WINDOWSはマイクロソフト社の登録商標です

応募シール
S・アスキーII

【販売代理店】●Computer Land 第一部 ☎03-615-3143●J&P U.S.LAND ☎06-634-1411●㈱PCテック ☎03-5256-0181●ロビン電子㈱ ☎03-255-6027●SPA㈱ ☎03-293-7191●T-ZONE ☎03-257-2652●㈲丹青通商 ☎03-254-5730●㈱巴商会 ☎03-663-2957●WORLDSOFMAP ☎03-5256-2926●㈱アトム ☎03-951-8181

NBCC 日本ボードコンピュータ株式会社
〒110 東京都台東区台東1-1-5 ANTEX24ビル
☎03-837-5866(代) FAX 03-837-5867

資料請求 No.179

Cのベストパートナー

●ソースコード付属／組み込みロイヤルティ無料●

The C Libraries

CCF／総合コンソールコントロールライブラリ　価格 ¥59,800
CCF／EX　価格 ¥98,000

キーボード、テキスト画面、グラフィック、マウス、ウィンドゥ、メニュー、プリンタ、拡張関数等200以上の関数で構成される総合コンソールコントロールライブラリです。『Graphic & Mouse』、『Window & Menu』、『Key & Display』とプリンタ関数、拡張関数で構成されています。

■『Graphic & Mouse』、『Window & Menu』、『Key & Display』については各製品の説明をご覧ください。　■プリンタ関数はBIOSを介してプリンタに文字コード、制御コード送り、各種アルファベットの文字印字、拡大、キャラクタモードの切り替え、フォームフィード、プリンタ状態の設定／検査、印字方向の切り替え、サブスクリプト／スーパースクリプト文字の印字、縦印字、強調、アンダーライン、ドラフトモードの切り替え、左右マージンの設定、改行幅／方向の切り替え、ソフトウエアリセット等の処理を行うことができます。対応プリンタはPC-PR系です。　■拡張関数はMS-DOSファンクションコールを用いたキー入力関数、文字列処理関数、y(es)／n(o)の入力を促し、Shift、カナシフト、全角、半角のいずれの状態でもy／nを判断する関数等があります。

◆『CCF／EX』ではXL、XL^2、RLのハイレゾリューション／ノーマルの両モードに対応し、一部の関数を除いて、どちらのモードでも動作するプログラムを開発することができます。

プリンタ関数(抜粋)

文字出力／プリンタステータスの取得／プリンタの初期化／プリンタステータスの取得／CR、LF出力／漢字文字出力／ANK文字列出力／漢字、ANK文字列印字／漢字、ANK文字列印字(ANK－>半角変換／漢字／ANK文字列縦印字(ANK－>半角変換)／プリンタのリセット／SELECT、DESELECTの切り替え／ドラフトモードの設定、解除／片方向、両方向の切り替え／改行方向の切り替え／データのキャンセル／フォームフィード／n/120インチ改行に設定／1/6インチ改行設定／1/8インチ改行設定／n回改行／漢字モード設定／解除／ANKモードの選択、設定／HS PICAモード設定／Condenseモード設定／Eliteモード設定／Proportionalモード設定／拡大倍率の設定／強調モード設定、解除／アンダーラインモード設定、解除／漢字縦モード設定、解除／漢字印字幅の変更／横倍角印字モード設定、解除／1バイト文字印字モードの変更／スクリプトモードの設定、解除／漢字半角モードの変更／ドットスペースの出力／左、右マージンの設定

Window & Menu　価格 ¥29,800
Window & Menu／EX　価格 ¥49,800

テキストウィンドゥ、メニュー関数のライブラリパッケージ。最近のちょっとしたソフトでは大抵ウィンドゥ機能を備えています。例えばスクリーンエディタでは画面を分割し2つ以上のウィンドゥを開いて編集できるとか、データベース、業務用ソフト等では簡単な説明や操作ガイドをヘルプウィンドゥを開いて表示したりしています。このような表示を行うためのマルチウィンドゥ機能とウィンドゥを利用したメニュー選択機能を提供します。

『Window & Menu／EX』はXL、XL^2、RLのハイレゾリューション／ノーマルの両モードに対応し、互換モード(自動判別)によりどちらのモードでも動作するプログラムを開発することができます。また、ハイレゾリューションモード時には31行／25行モードを選択することができます。

■ウィンドゥ／メニューウィンドゥあわせて20個までのウィンドゥがオープンできる（増減可能）■オーバーラップ／切り替え機能　オーバーラップにより隠れたウィンドゥを一番上に表示　■移動機能　ウィンドゥを指定された座標へ移動　■文字列表示　行を指定する関数と、上下スクロールを伴った文字列表示機能がある　■スクロール　スクロール関数と、文字列表示を伴った上下スクロール機能がある　■VRAMをアクセスしてウィンドゥ操作を行っているので高速。ウィンドゥの切り替えもほぼ瞬時　■ウィンドゥを開き、y(es)、n(o)の入力を促す関数と上下左右スクロール機能を持った文字列表示関数によるテキストファイル等のスクロール表示が可能な拡張ウィンドゥ機能　■カスタマイズド・ウィンドゥ　ウィンドゥのフレームキャラクタ、フレーム色、ウィンドゥ内の表示色の設定が可能です。この機能により、好みの色、キャラクタを用いたウィンドゥを定義可能　■横長タイプ、縦長タイプ、羅列タイプの3タイプのメニューウィンドゥにより、用途に応じたメニュー表示が可能　■サンプルプログラム　メニューウィンドゥにファイル一覧を表示し、選択されたファイル内容を上下左右スクロール機能を持った文字列表示関数により、スクロール表示をおこなうサンプルプログラムが付属。

次の関数の他、『Key & Display』の＊印が付いている関数も加わります。

関数(抜粋)

ウィンドゥの初期化、終了処理、オープン、クローズ／すべてのウィンドゥをクローズする。表示対象となっているウィンドゥをクローズする／表示対象ウィンドゥを別のウィンドゥに切替える／ウィンドゥの移動／ウィンドゥ内をブランク文字で消去／ウィンドゥ内に文字を表示、上へスクロールし、文字列を表示／下へスクロールし、文字列を表示／ウィンドゥを上、下にスクロール／ウィンドゥのフレームタイプを登録／文字列を指定行表示／ウィンドゥ内に文字列を表示し、キー入力によりテキストをスクロール／ウィンドゥを開き、'y'または'n'キーを入力を促す／水平メニューウィンドゥをオープン／表示し、選択を行う／ポップアップメニューウィンドゥをオープン／表示し、選択を行う／ワイドメニューウィンドゥをオープン／表示し、選択を行う／H.ハイレゾリューション／ノーマルモード、25／31行モードの設定を行う(H.マークが付いている機能はハイレゾモードでのみ使用可能)

Graphic & Mouse　価格 ¥29,800
Graphic & Mouse／EX　価格 ¥59,800

N88BASIC相当のグラフィック機能とマウス機能を提供するライブラリパッケージです。

グラフィック機能はグラフィック画面に点、線、四方形、円、文字等を描画する機能、指定領域を切り出す／貼り戻す等の機能と表示モードの設定を行う制御機能で構成されています。

マウス関数にはマウスカーソルの表示／非表示、色指定、ボタンの押下状態の検査、移動範囲の指定、移動距離の検査等の関数があります。

■N88BASIC相当のグラフィック関数を提供　■高速グラフィック関数　高速線分描画、高速連続線分描画、高速円描画、グラフィック画面への文字表示関数を備えています。(ノーマルモードのみ) これらの関数はすべてREPLACE／CLEAR／OR／XORの4つの描画モードのいずれかを選択指定することができます。(fline, flines, fcircle, g_char) ■マウス関数

◆『Graphic & Mouse／EX』ではXL、XL^2、RLのハイレゾリューション／ノーマルの両モードに対応しており、どちらのモードでも動作する互換モード(自動判別)もあります。

関数(抜粋)

画面のモード設定／表示範囲の設定／画面色の設定／パレットの設定／グラフィック、テキスト画面消去／ドットセット、リセット／直線、円、円弧の描画／ペイント／タイルパターンによる塗りつぶし／描画情報の取得、貼り戻し／スクロール／ドットのパレットコードの検査／ハードコピー／表示範囲の検査／日本語全角文字列描画／半角を含む日本語文字列描画／H.互換モード設定／H.簡易スタート処理／N.ドットパターンの描画／N.円描画／N.線／四辺形描画／N.連続線描画／マウスカーソルの表示、消去、座標を取得、設定／左、右ボタンの押下情報の取得／カーソル移動範囲の設定／カーソルの形状、色の設定／移動量の取得／ミッキー、ドット比の設定(N.はノーマルモード、H.はハイレゾリューションモードのそれぞれ専用関数です)

RS-232Cライブラリ　価格 ¥49,800
RS-232Cライブラリ／EX　価格 ¥98,000

本製品はC言語からRS-232Cインターフェースを介して、各種周辺装置、コンピュータ等とコミュニケーションをとるためのライブラリです。

パッケージにはBIOSを呼び出して行っている『RS-232Cライブラリ』、これに9861K用とマルチサーバ用デバイスドライバ版を追加拡張した『RS-232Cライブラリ／EX』の2種類があり、呼び出し方法は共通になっています。

『RS-232Cライブラリ』は主にBIOSを呼び出して、RS-232Cのコントロール、送受信を行っています。もちろん、2、3回線もサポートしています。

『RS-232Cライブラリ／EX』ではBIOS呼び出し版の他に、使用環境に制限がなく本体はデバイスドライバ化され、インターフェースライブラリを介して呼び出すデバイスドライバ版とマルチサーバ対応のデバイスドライバが拡張されています。また、各ライブラリ特有の関数(一部)を除いて上位関数の入口名、引数は共通なので、リンクするライブラリファイルを変更するだけで切り替えることができます。この他、オートダイアル／モデムコントロール関数、時間関係関数、および簡易ターミナルエミュレータ、XMODEM送受信プログラム等がサンプルプログラムとして提供されています。製品にはデバイスドライバを含むすべてのソースコードが含まれています。

■用途　回線数によって使い分けのできる3種類のライブラリ(EX)を装備。　■マルチサーバボードを装着することにより9回線までサポート可能(EX)　■『RS-232Cライブラリ』では"Ver. 3.1以上かつ型番がPS98-127-??? 以降のMS-DOSでなければならない"という制限があります。

関数(抜粋)

■共通　回線の選択、初期化／受信バッファ内データ長の検出／受信バッファ内のデータを得る／1文字出力／8251にコマンドを送る／8251ステータスを得る／送信レディ状態の検査／キャリアディテクトの検出　RS-232Cインターフェースのリセット／文字列入力／文字列出力／1文字受信／1文字送信／漢字処理を伴うデータ送信／モデムの初期化、接続、自動発信
■EX専用　ブロック受信／ブロック送信／ハンドルのクローズ

資料請求 No.180

BGM／グラフ作成プログラム　価格 ¥98,000
BGM／EX　価格 ¥148,000

12種類のグラフ作成機能を提供するパッケージです。例えば、必要なパラメータを設定し、関数を呼び出すだけで円グラフや棒グラフを簡単に描画することができます。さらに、タイルパターン、ラインパターンとの対応、日本語コード文字列をグラフィック画面に表示する機能をサポートしているので、ビジネスアプリケーションで必要とされるあらゆるグラフ情報を表現できます。

■必要なパラメータを設定し、関数を呼び出すだけで簡単にグラフを描画できます。サンプルプログラムを見ればすぐ理解できます。円グラフの例　cg(vlist,vcnt,cx,cy,r,scalel,scale2)；　■12種類のグラフ作成機能　■描画位置、大きさを好みに合わせて描画できる　■サポート関数を用意　タイルパターン(塗りつぶしパターン)、ラインパターンの一覧を表示する関数により、例えばタイルパターンとその内容(例えば"商品名")との対応を簡単に表示できます。　■多彩な表現が可能　円グラフ、ドーナッツグラフ、パイチャート、帯グラフでは%または数値の表示／非表示の選択が可能、縦棒グラフ、横棒グラフ、折線グラフではスケールの表示／非表示の選択が可能、タイルパターン、ラインパターンと表示色の組み合わせ、ボーダーカラー(円、棒等の外枠の色)、スケールカラー、%または数値の表示色の指定が可能なため、多彩な表現が可能です。　■すぐ使える、拡張もできる　24種類のタイルパターン、10種類のラインパターンを登録済み、さらにユーザー登録も可能　■ハードコピー　グラフィック画面、テキスト画面、グラフィック画面とテキスト画面の重ね合わせが可能。日本語コード(Shift-Jis：全角、半角、1／4角)文字列をグラフィック画面に表示できるので、ガイドメッセージ等をテキスト画面に表示したままの状態でもグラフィック画面だけのハードコピーを印刷することができます。

◆『BGM／EX』ではXL、XL²、RLのハイレゾリューション／ノーマルの両モードに対応しており、さらにどちらのモードでも動作する互換モード(自動判別)もあります。

以下の関数に『Graphic & Mouse』または『Graphic & Mouse／EX』の一部の関数が含まれます。

円グラフ／ドーナッツグラフ／パイチャート／横棒グラフ／横棒グラフ(複数)／横棒グラフ(積み重ね)／縦棒グラフ／縦棒グラフ(複数)／縦棒グラフ(積み重ね)／帯グラフ／折線グラフ／レーダーチャート／ラインパターンリストの表示／タイルパターンリストの表示／タイルパターンの作成

Key & Display　価格 ¥29,800

Key & Displayはソフト開発で欠かせないファンクションキー及びANK、漢字の入力、VRAMを直接アクセスしての文字表示等100を越える関数で構成されています。ファンクションキー自身には一切特別な値や文字列を設定せずに、BIOSレベルの処理で行っているため、どのレベルの関数でも同じ処理でファンクションキーの取得も可能です。また、VRAMを直接アクセスして表示するので25行目に文字列を表示してもスクロールしてしまうことはなく、すべての1バイトキャラクターコード(0×01−0×f7)を表示することができます。文字表示、文字属性の設定(色／属性)、VRAM上の指定領域を切取り、貼り戻し、表／裏画面の切り替え表示、全画面／部分スクロール、ファンクションキー表示等の機能を持っています。

■キー入力　ファンクションキー及びANK、漢字の入力。ファンクションキーとSHIFT、CTRLキー等のシフトキーとの組合せで、別のキーコードを得ることもできます。　■日本語入力フロントエンドプロセッサ(FEP)のコントロール　日本語入力FEPのON／OFF、入力モードの切り替え等のコントロールができます。対象となる日本語入力FEPはATOK4／5／6／7(6モード)、VJE-α／βです。　■漢字の入力はFEP等を介して行なわれ、2バイトで返されます。　■VRAMを直接アクセスして文字表示を行っているので非常に高速です。　■すべての1バイトキャラクターコード(0×01−0×f7)を表示することができます。　■全画面に表示可能　25行目に文字列を表示してもスクロールしてしまうことはありません。　■裏画面を有効に利用できる　表／裏画面を切り替えて表示選択できるのでスピードアップ、VRAMの有効利用が可能です。　■画面の指定領域を切取ったり／貼り戻したりが簡単にできます。　■画面上に表示されている内容をそのままファイルにセーブしたり、ファイルにセーブされた画面内容を再表示することが可能です。セーブ／ロードする領域を指定することができるので全画面／部分のどちらも可能です。　■すべての文字属性(シークレット、ブリンク、リバース、アンダーライン、バーチカルライン、色指定)を画面上の指定領域に対して設定できるので表現力豊かな画面表示が可能です。　■多彩な画面操作機能を装備　画面上の指定領域に対する文字／文字属性のコピー、移動、消去を行う関数が用意されています。　■画面上の指定領域に対する上下左右スクロール機能により、全画面／部分のスクロールが可能です。　■現在表示されているすべての1バイトキャラクタコード／漢字の文字列をVRAMから取り出ししASCII／Shift-Jisコードへの変換が可能です。　■指定座標の表示文字が漢字か否か、また漢字の2バイト目か等の識別機能により、現在表示されている文字を検査することができます。　■カーソルコントロールシステム　2種類のカーソル移動関数群とカーソル移動範囲限定機能により、多彩なカーソルワークが可能になりました。通常の上下左右移動、座標指定移動、カーソル位置取得機能はもちろん、表示文字列のコード識別機能を使用したカーソル移動関数群を用意しました。この関数群は移動位置が漢字の2バイト目か否かの検査を行い、移動位置が漢字の2バイト目の場合はスキップし、次の位置へ移動する関数群です。カーソル移動範囲を一定領域に限定する関数も用意しました。■まだまだある、こんな機能、こんな関数　指定文字／文字属性によるVRAMの消去関数、指定座標からn文字を消去する関数、カーソルの表示／非表示切り替え、カーソルブリンクのON／OFF、ファンクションキー表示のセーブ／リストア／設定／消去etc…

関数(抜粋)

ATOK、VJEのコントロール／ファンクションキーの退避、設定／＊漢字、ファンクションキーを2バイトコードで返すキー入力／全角、ANK文字キー入力／＊ファンクションキーの直接入力／＊テキストを箱型に消去／＊テキストの指定範囲のイメージの取得、貼り戻し／テキストから箱型の文字列をコピー、移動／＊テキストから箱型に文字属性を取得、貼り戻し／＊カーソルを指定座標に移動／＊カーソル座標の取得／カーソルを移動範囲内で移動／カーソルを上下左右に移動／カーソルホームポジションを設定／カーソルをホームポジションへ移動／全部、部分、文字単位での消去／ファンクションキー表示の退避、再設定、表示、消去／＊テキストにASCII、全角、文字列を直接表示／指定されたバイト数分文字列を表示／現在のカーソル位置に文字列を直接表示／＊表示文字が漢字か否かの検査／テキストの文字、文字列コードの取得／カーソルブリンク、表示のON、OFF／＊テキストの切り替え／CRTのモードの検査、変更／スクロールし、文字列を表示／＊上、下、左、右スクロール(日本語入力FEP制御関数はノーマルモードでのみ使用可能。＊は『Window & Menu』および『Window & Menu／EX』でも提供)

CodeBase4　価格 ¥98,000

CodeBase4はC言語からdBASEⅢ、dBASEⅢplus、dBASEⅣ、Clipperのファイルシステム(データベース、インデックス、メモ)とコンパチブルなデータベースライブラリです。これらのデータベース言語にコンパチブルな関数を用意しているのでdBASEⅢ/plusからの移植も簡単にできます。dBASEⅣと比較してデータアクセスは5倍速く、ループではなんと330倍も速くなります。

■高速処理　■最大1,022フィールド、2,000,000,000レコード(論理値)　■メモリモデル　S(一部機能が限定)／M／L　■2,000行程度のソースプログラムで100kb程度の.exeファイルを作成可能　■マルチユーザー対応　■PC-9800シリーズではデータベースライブラリで漢字の使用は可能　■IBM-PC版ではウィンドゥ、メニュー関数も付属　■英文マニュアル　◆対応C言語：Microsoft-C 4.0／5.1またはTurbo-C2.0

関数(抜粋)

■[block]　ブロックキーを追加、削除／ブロックを検索／キー番号のスキップ　■[conversion]　指定長のASCII文字列をdouble／int／longに変換　■[database]　データベースファイルの作成、オープン、選択／指定レコードに削除マークを付加／削除マークが付加されているかの検査／レコードを読む／ロック／削除マーク付きのレコードを削除／レコード数、番号の取得／参照番号の取得／データベースの選択／インデックスファイル内の先頭レコードを読む／カレントデータベースのロックの解除／データベースのオープン、選択／データベースを選択後、ロックする／レコードを書き出す／指定された範囲のレコードの削除　■[expression]　dBASEの式を評価し、結果を得る／dBASEの式を中間コンパイルする／演算子の優先順位の取得　■[field]　10進数の桁数の取得／フィールドリファレンス番号の取得／フィールド名の取得／カレントデータベースのフィールド数の取得／レコードバッファの先頭ポインタの取得／レコードバッファ内の文字の総数の取得／フィールドリファレンス番号の取得／論理フィールドがTRUEかFALSEかの検査／指定フィールドの型の取得／数値フィールドの値をdouble得る／指定フィールドのバイト数の取得　■[index file]　インデックスファイルを作成、オープン、クローズ、ロック、ロックの解除、選択キーの追加／ポインタを最後尾に移動、指定キーに設定／ファイルリファレンス番号の取得／インデックスし直す／キーをサーチ／ポインタを先頭へ設定／インデックスファイルの型を文字で返す／ロックの解除　■[memo]　内部的にデータが格納されているかを検査／メモファイルにある情報を書き出す／メモフィールドを読む、データが存在するか調べる　■[utility]　次の該当ファイル名を探す／セクションのロック、解除／ファイル名に適した文字か調べる／ファイル名と拡張子を結合する／ファイル名から拡張子、ディレクトリ名を取り除く／ファイルをオープンする　■[extended]　レコードがブランクかどうか調べる／メモリバッファに入るだけのレコードをシーケンシャルに読む／レコードバッファ内のレコードをn番目とする／データベースの内容を出力デバイスに書き出す／関連データベースと管理データベースを関連付ける／フィールドリファレンス番号によって指定されたフィールドの総計を得る

●まだまだあるこんなライブラリ

■Clisp	—Lispライク関数ライブラリ—	価格 ¥98,000
■AX-WINDOW	—AXマシン用テキストウィンドゥ／メニューライブラリ—	価格 ¥98,000
■FMR-WINDOW	—FMR-50／60用テキストウィンドゥ／メニューライブラリ—	価格 ¥98,000
■FMR-GRAPHIC	—FMR-50用グラフィックライブラリ—	価格 ¥98,000
■Ptree & Symtab	—構文解析／シンボルテーブルモジュール—	価格 ¥69,800
■CCF／総合コンソールコントロールライブラリ for FORTRAN		価格 ¥79,800

※すべての表示価格には消費税は含まれておりません。

●走行環境等

●機種……NEC PC-9800シリーズ(LT、ハイレゾリューションモードを除く)
『CCF／EX』、『BGM／EX』、『Graphic & Mouse／EX』、『Window & Menu／EX』はXL、XL²、RLのハイレゾリューション／ノーマルの両モードに対応。
●MS-DOS…Version 2.11／3.1／3.3
●対象C言語…Microsoft-C 4.0／5.1　Quick-C 1.1／2.0　Lattice-C 3.1／4.0　Turbo-C 2.0
●メディア……5" 2HD、3.5" 2HD、5" 2D (IBM-PCフォーマット)

お求め方法

●お近くのパソコンソフト取扱店または弊社までご注文ください。
●ご注文の際に製品名、対象コンパイラ(ヴァージョン)、メディアを指定してください。

クイック資料送付サービス

●いますぐ、資料をご希望の方はお電話ください。FAXで資料をお送りいたします。

TIPS 株式会社ティップス
●〒111 東京都台東区蔵前2-8-10
●PHONE 03-866-8635 ●FAX 03-866-8637

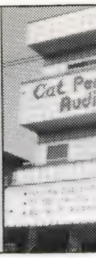
パソコン＆AV専門店

キャット

通信販売 受付時間 AM11:00～PM19:00（木曜日定休）

TEL.0482-65-6531

埼玉県川口市芝2-2-7　⊘はメーカー希望価格です。



NEC	エプソン	東芝	富士通	シャープ
PC-9801N 98ノート ⊘¥248,000 **¥139,800**	PC-CLUB ⊘¥168,000 TELにて 特価販売中	J-3100SS001 ⊘198,000 **¥108,000**	FM TOWNS 2F 本体JISキーボードテンキー付、ディスプレイ、OSセット ⊘¥517,800 **¥290,000**	CZ-623C-TNチタン ⊘¥498,000
PC-9801NS 98ノートS ⊘¥298,000 **¥192,000**	PC-286VG ⊘¥268,000 TELにて 特価販売中	J-3100SS002 ⊘¥228,000 **¥158,000**	FM TOWNS 2H 本体JISキーボードテンキー付、ディスプレイ、OSセット ⊘¥687,800 **¥415,000**	CZ-603C ⊘¥338,000
PC-9801RX21 ⊘¥338,000 **¥208,000**	PC-386M ⊘¥328,000 TELにて 特価販売中	J-3100SS02E ⊘¥328,000 **¥200,000**		CZ-613C ⊘¥448,000
				CZ-653C ⊘¥285,000
				CZ-663C ⊘¥398,500
				大特価販売中 TELにて

パソコン本体

品名	⊘価格	特価
PC-9801RA21	⊘¥498,000	**¥318,000**
PC-9801RA51	⊘¥668,000	**¥400,000**
PC-9801TW2	⊘¥548,000	**¥359,000**
PC-9801TW5	⊘¥758,000	**¥495,000**
PC-9801TS5	⊘¥850,000	**¥558,000**
PC-9801RS21	⊘¥398,000	**¥268,000**
PC-9801RS51	⊘¥568,000	**¥373,000**
PC-9801NS-H20	⊘¥448,000	**¥339,800**
PC-286UX-STD	⊘¥268,000	
PC-386VR-STD	⊘¥348,000	
PC-386M-H40	⊘¥448,000	
PC-386V-STD	⊘¥498,000	
PC-286NOTEf	⊘¥198,000	
PC-86BOOK	⊘¥258,000	
PC-286BOOK-H20	⊘¥378,000	

大特価販売中

NEC PC-8801MC2
⊘¥199,000
限定30台
¥105,000
注目新品 特価

プリンター

品名	⊘価格	特価
PC-PR201J	⊘¥248,000	**¥138,000**
PC-PR201GS	⊘¥135,000	**¥ 89,800**
PC-PR201TH	⊘¥138,000	**¥ 98,000**
PC-101GS	⊘¥118,000	**¥ 79,800**
PC-PR101TL3	⊘¥ 69,800	**¥ 46,000**
PC-PR102TL3	⊘¥ 50,000	**¥ 33,800**
PC-PR150VH	⊘¥ 99,800	**¥ 67,800**
VP-2050PC	⊘¥158,000	
VP-1500PC	⊘¥138,000	
VP-950PC	⊘¥128,000	
VP-550PC	⊘¥ 87,000	
AP-850PC	⊘¥ 96,800	
AP-550EXPC	⊘¥ 69,800	
HG-4800PC	⊘¥348,000	**¥158,000**
キャノン BJ-130J PCセット	⊘¥198,000	¥ TELにて

間違いなく特価

ハードディスク・フロッピードライブ

メーカー	品名	⊘価格	特価
コンピュータリサーチ	CRC-MH4B	⊘¥ 99,800	**¥ 58,000**
	CRC-MH8B	⊘¥158,000	**¥ 99,800**
テクノジャパン	PD-50RS	⊘¥118,000	**¥ 86,800**
	PD-100RS	⊘¥168,000	**¥119,800**
ICM	SR-80	⊘¥168,000	**¥119,800**
緑電子	3.5″1Dリトルf	⊘¥ 39,800	**¥ 24,800**

ディスプレイ

品名	⊘価格	特価
PC-KD854N	⊘¥ 84,800	**¥ 42,000**
PC-KD853N	⊘¥118,000	**¥ 50,000**
PC-TV454	⊘¥128,300	**¥ 74,800**
PC-TV454	⊘¥147,000	**¥ 84,800**
PC-TV354	⊘¥ 99,800	**¥ 64,800**
PC-KD881	⊘¥118,000	**¥ 78,000**
PC-KD882	⊘¥ 89,800	**¥ 58,000**
Cu-14FD	⊘¥ 74,800	**¥ 42,000**
Cu-14KD	⊘¥ 89,800	¥ TELにて
XC-1498CII	⊘¥107,000	**¥ 50,000**
CMT-147H	⊘¥ 79,800	**¥ 43,000**

各社特価ワープロ

富士通 オアシス30LXII ⊘¥188,000 **¥118,000**	富士通 オアシス30AX ⊘¥228,000 **¥123,000**
東芝 ルポ95HP ⊘¥248,000 **¥168,000**	東芝 ルポ95FG ⊘¥198,000 **¥123,000**
NEC PWP-70R ⊘¥198,000 **¥118,000**	NEC PWP-70RA ⊘¥278,000 **¥180,000**
パナソニック FW-UIP503AIX ⊘¥175,000 **¥ 89,800**	キヤノン BJワープロα-50 ⊘¥198,000 特価
カシオ電子手帳 DK-5000 ⊘¥ 28,000 **¥ 15,000**	NEC PC電子手帳 PC-ETI ⊘¥348,000 **¥ 26,800**

通信販売

お申し込みは簡単、今すぐお電話で!!

1. まずはお電話0482-65-6531へ
2. 商品名、型番を担当へ!!また発送指定日をどうぞ!!
3. 当店指定口座へお振込お願いします。
4. 確認後、商品発送➡お客様の手へ

※一部の商品に限り納期が2日～10日間かかる時があります。

指定口座/三和銀行わらび支店(普)434123 ㈱キャット宛へ

好評受付中!

情報サービス始めました!! キャット特別価格をFAXにて案内!!

※AV商品・家電製品の価格情報を定期的にご案内致します。
※団体会社、学校関係を原則とするお手持ちFAX機に、月1回、無料案内!!

お申し込みはFAX0482-65-6731まで

❶ 全品メーカー保証付。
❷ 配送料は1ケ口620～1,620円。(地域により異なります。)
❸ 銀行振り込みはお客様負担です。

本紙の表示価格には消費税が含まれておりませんので、表示価格＋3％の売価になります。価格は変動することがありますのでTELにてご確認下さい。

ASCII HOT LINE

TAKE OFFキャンペーン御礼

'90年7月1日よりスタートさせていただきました『TAKE OFFキャンペーン』アスキーセンセーション1も9月30日をもちまして、大好評のうちに幕を閉じさせていただきました。ご応募をいただきました皆様に紙面ではございますが厚く御礼申し上げます。また、第2回目のご当選者のお名前は、12月号の当ページにて発表させていただきます。

アスキーセンセーション2『アスキー・パーソナルCAD大賞』の作品募集は'90年10月31日迄となっております。腕を振るってのご応募をお待ち申し上げております。

株式会社アスキー　営業本部　TEL.03-486-8080

『ESterm 2』ご利用のお客様へ

『ESterm 2+』(イーエスターム2プラス) 近日発売のお知らせ

'89年3月に発売以来、多くのネットワーカーの方々にご好評をいただいております『ESterm』が、最新プロトコルをサポートしてより使いやすく、より先進的に変身いたしまして新登場します。

『ESterm 2+』での新機能

- ●最新のプロトコルを搭載
  ◎B Plus：NIFTY-Serveの高速ファイル転送プロトコルを搭載。
  ◎ZMODEM：アスキーネットの高速ファイル転送プロトコルを搭載。
  どちらのプロトコルもMS-DOSモードではなく『ESterm 2+』の中から直接起動でき、ファイルの自動受信などができます。
- ●VJE-β Ver. 2.5を標準搭載、先進のワープロ機能
  最新のかな漢字変換プロセッサVJE-β Ver. 2.5を利用して、通信中に文章を編集することができます。
- ●ファイルチェッカーなど、パソコン通信に必要なPDSを添付。
  - ●ファイルチェッカー
  - ●ファイル圧縮プログラム
  - ●バイナリ／テキストファイルコンバーター
- ●対応機種
  (NEC)PC-9800シリーズ
  9801／E／F／M／U／VF／VM／UV／VX／UX／CV／LV／RA／RX／EX／ES／LX／LS／T
  PC-9801N(98NOTE)／NS(98NOTE SX)
  PC-98XA／XL／XL2／RL／H98(ハイレゾモード対応)
  ＊PC-9801／Eでは漢字ROMボードが必要です。

『ESterm 2』ご購入のお客様で、(株)アスキーへのユーザー登録を完了されております方には、バージョンアップの詳細のご案内を、準備ができ次第発送させていただきます。まだ、ユーザー登録カードをお手元にお持ちのお客様は、ご返送をいただけますようお願い申し上げます。

『アスキーカード』新規会員募集のお知らせ

アスキーカードは、皆様に"充実したパソコンライフ"をお送りいただくためにお届けいたしますクレジットカードです。アスキー製品、ASCII-NETにはもちろん、全国約32万の加盟店を有する日本信販クレジットカードとして、170ヵ国約610万の加盟店を持つVISAカードとしてもご利用いただけるカードです。

●インフォメーション
会員の皆様に新鮮な情報、お得な情報を満載したリフィルサイズの会報誌を毎月発行いたしましてお手元にお届けいたしております。また、パソコン関連のショウ、イベントなどのご案内を随時いたします。

●『体験ディスク』サービス
アスキービジネスソフトの機能、内容を確認してから購入したい。そんな会員の皆様のご希望にお応えいたしまして、機能を制限したシステムとサンプルを収録いたしました『体験ディスク』をご要望に応じまして随時差し上げております。

●アスキーカードパソコン保険
アスキーカード会員の皆様の大切なパソコン本体、周辺機器、ソフトを火災、落雷や盗難、取り扱い上の不注意などさまざまな事故からガードするアスキーカードならではのサービスです。

●ASCII-NET
アスキーのパソコン通信サービス『ASCII-NET』が基本料金のみでご利用いただけます。また、全国の主要都市にアクセスポイントのある『アスキーTri-Pネットワークサービス』もご利用いただけます。

アスキーカードは、18歳以上のお勤め(自営)の方、主婦の方、学生の方(高校生以下はご入会いただけません)にご入会いただけます。

詳しい資料、ご入会をご希望の方は、下記までお電話か、おハガキにてお問い合わせください。折り返し、ご入会の詳しい資料、お申込用紙をお送りさせていただきます。

〒107-24 東京都港区南青山6-11-1
株式会社アスキー　アスキーカード事務局　TEL.03-797-7601

アスキービジネスパソコンスクール 11月度日程

アスキーのビジネスソフトをより活用いただくために、東京・大阪の2会場で下記の日程でパソコンスクールを開催いたしております。

コース名	日　　程	料　金
3·D PERS紹介コース	12月12日(水)(10:00-12:00)	無　料
パソコン入門	11月1日(木)(10:00-12:00) 12月4日(火)(10:00-12:00)	3,090円(税込)
CARD3 プリント	12月7日(金)(10:00-12:00)	
The CARD3入門	10月24日(水)(10:00-17:00) 11月2日(金)(10:00-17:00) 11月20日(火)(10:00-17:00)	10,300円(税込)
The CARD3実習	11月9日(金)(10:00-17:00) 12月11日(火)(10:00-17:00)	10,300円(税込)
The CARD3応用	10月30日(火)(14:00-17:00) 11月28日(水)(14:00-17:00)	5,150円(税込)
The CARD3システム設計	12月18日(火)(10:00-17:00)	20,600円(税込)
Recalc入門	11月13日(火)(14:00-17:00)	5,150円(税込)
パソコン通信入門(東京会場のみ)	11月14日(水)(14:00-17:00)	
CANDY4入門	10月31日(水)(14:00-17:00) 11月22日(木)(14:00-17:00)	
Z's WORD JG2入門	12月19日(水)(14:00-17:00)	

◆各コース共、定員10名となっております。先着順にて受け付けさせていただいておりますので必ずお電話にてご予約ください。
◆受講料金のお支払いにアスキーカードをご利用いただきました場合は、受講料金を1割引とさせていただきます。
◆詳細につきましては下記教室係までお電話にて直接お問い合わせいただけますようお願い申し上げます。
◆アスキービジネスパソコンスクール以外でも、アスキー製品の講習会を受講することができます。東京・大阪地区以外で受講ご希望の方もお気軽にお問い合わせください。

[東京教室] 〒107-24 東京都港区南青山6-11-1 (株)アスキー 教室係 電話03-486-4511
[大阪教室] 〒530 大阪市北区梅田1-11-4 大阪駅前第4ビル (株)アスキー 教室係 電話06-348-0005

アスキーに対するご意見、ご希望、また弊社の販売についてお気付きの点などがございましたら、(株)アスキー営業部「HOT LINE」係宛ハガキにてお送り下さい。よろしくお願いします。

製品についてのお問い合わせは以下の通りです。

＊出版物	486-1977
＊ソフトウェア	486-8080
＊ファミコン	486-5137
＊アスキーネット関連	486-9661

製品のお買い上げ後のご質問はユーザーサポート宛

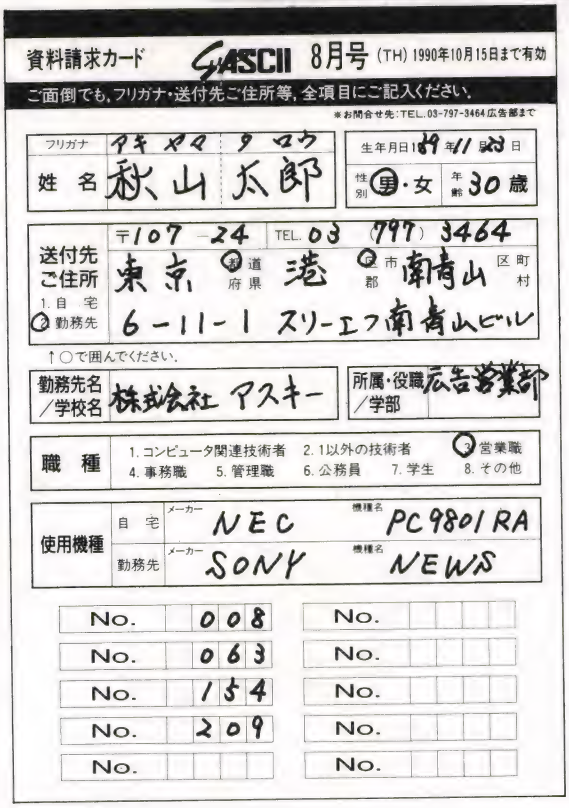
＊出版物＊ゲーム(ファミコンを除く)＊通信ソフト	498-0299
＊ビジネスソフト	498-0205
＊アスキーネット関連	486-9661

〒107-24 東京都港区南青山6-11-1スリーエフ南青山ビル　**株式会社アスキー**

ア ㈱アイザック……表3
㈱アイシーエム……10
カ キャット……182
キヤノン㈱……4、5
クボタコンピュータ㈱……表4
サ ソニー㈱……表2、1
タ ㈱ティップス……180、181
ナ 日本鋼管㈱……12、13
㈱日本ボードコンピュータ……179
日本サンマイクロシステムズ㈱……6、7
日本システム㈱……177
日立造船情報システム㈱……00A
ハ ㈱PFU……11
フジキンソフト㈱……14、15
富士ソフトウェア㈱……16、178
ヤ ㈱横河ヒューレットパッカード……8、9

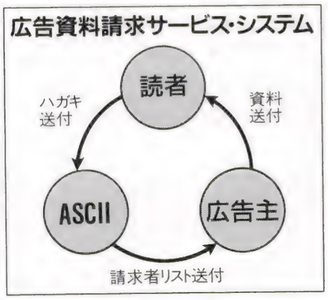
Super ASCIIの「広告資料請求サービスシステム」

簡単・便利で手間いらず



ぜひ、ご利用ください。

Super ASCIIに掲載された広告製品、サービスに関する資料集めは「広告資料請求サービスシステム」をご利用ください。本誌に添付された資料請求カードをアスキーに返送くださるだけで、ご希望の資料はすべて入手可能です。ぜひ、有効にご活用ください。



- 各広告の下にある資料請求番号を「資料請求No.」の欄にご記入ください。
- 資料請求カードは、ご面倒でも全項目にご記入ください。
- 資料請求カード以外によるご利用は、無効とさせていただきます。

★姓名のフリガナ、お電話番号、ご住所の番地は必ずご記入下さい。記入もれがある場合資料をお届け出来ない場合があります。

お問い合わせ先：〒107-24 東京都港区南青山6-11-1 スリーエフ南青山 株式会社アスキー 広告部 TEL.(03)797-3464

注：フリガナを忘れずお書き下さい。

本誌をご愛読いただきありがとうございます。
お近くに書店のない場合は、弊社に直接お申し込み下さい。金額の詳細につきましては書籍広告の頁を参照下さい。

切り取り線

切り取らないで郵便局にお出しください。

この欄は、加入者あての通信にお使いください。注：フリガナを忘れずお書き下さい。

お届け先	お名前		電話	自宅
				勤務先
	御住所	〒		

注文書	スーパーアスキー 定期購読	新規申込 年 月号より	継続：定期購読 No.	年間（12冊）8,160円（税込）
	UNIXマガジン 定期購読	新規申込 年 月号より	継続：定期購読 No.	年間（12冊）9,720円（税込）
	MSJ定期購読	新規申込 年 No. より	継続：定期購読 No.	年間（6冊）7,500円（税込）
	品名	円	〒 円	計 円
		円	〒 円	計 円
		円	〒 円	計 円
	合計金額			円
通信欄				

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

通常払込料金加入者負担	払込通知票		
口座番号	東京	4-161144	金額 ※ 億 千 百 十 万 千 百 十 円
加入者名	株式会社アスキー		料金 払込み 特殊 円
払込人住所氏名	※ (郵便番号 -) ご住所		備考
	お名前 (☎ - -)		受付局日附印

各票の※印欄は、払込人において記載してください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

切り取らないで郵便局にお出しください。

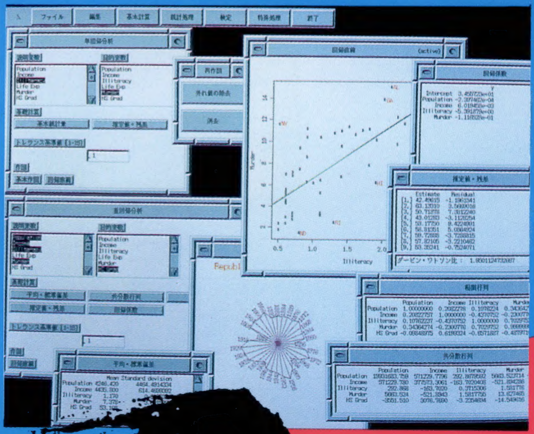
通常払込料金加入者負担	払込票	
口座番号	東京	4-161144
加入者名	株式会社アスキー	
金額	※ 億 千 百 十 万 千 百 十 円	
払込人住所氏名	※	
備考		受付局日附印

切り取り線

知的マン・マシン・システムを創造する

isac International Sekisui AI Corporation

米国AT&Tのベル研究所で開発され、理想的なデータ解析の環境を実現するSシステム。ウィンドウ・ベースの優れたインターフェイスにより、対話型で高度なデータ解析とグラフィック表示が行えます。また、「言語」と「支援システム」の両方の特徴を兼ね備え、統計調査や財務分析、経営管理、学術研究などのための計算・作図・データ解析など広範囲でご利用いただけます。



《対応機種価格》
ソニーNEWS(XII)……68万円
SUN3(XII)……68万円
SPARC……68万円
HP9000／300……68万円
OMRON／LUNA……68万円
(各機種とも、大学向け特別割引価格……24万円)
(※消費税別)

NEW

'84年リリース以来、1万以上のユーザを獲得してきた統計解析とグラフィックスのための対話型環境、"S"が大幅に機能強化されました。
「The New S Language」に基づいた新バージョンの"S"は、以下の特徴を持っています。

●S言語仕様が拡張され、オブジェクト指向を取り入れた高水準の関数型言語となっています。これにより、従来のマクロをSの関数として記述することが可能です。
●S関数はマクロの4～5倍高速で作動し、記述性も大幅にアップしています。
●関数、データ共に、「オブジェクト」として統一的に扱えるようになりました。
●UNIXインターフェースをさらに洗練し、C/Fortranで記述した外部関数との接続がより容易になりました。Cの関数中からSの関数を呼び出す事も可能です。
●実行時に必要なオブジェクトのダイナミック・ロード機能をサポート。また、倍精度演算、複素数、複素関数を新たにサポートします。
●UNIX上の標準ウインドウ・システムであるXバージョン11に対応しウインドウへの多彩なグラフ表示を始め、マウスによるグラフデータのポインティング、ポップアップ・メニューを扱うことができます。
●充実したオンライン・ヘルプ機能、ヒストリー機能等、対話性もさらにアップしました。
●漢字対応、漢字フォントも標準装備しています。

new S の 特 徴

①エラーメッセージやヘルプ機能が充実した対話型設計
②豊富なコマンド群を標準装備し、プログラミング不要
③既存のデータベースなどからデータ・コンバートが簡単
④強力なS言語により、独自の解析手法を登録可能
⑤多変量解析や時系列解析などの統計処理用関数を多数装備
⑥解析結果を30種以上の多様なグラフに表示可能
⑦各種プロッタ他、Xウィンドウ、POSTSCRIPTをサポート
⑧解析データの構造や属性等について細かい配慮不要
⑨漢字対応により、漢字等を含むグラフ表示可能

統計解析とグラフィックスのための対話型環境

S

●X Window Systemは、マサチューセッツ工科大学の商標です。
●POSTSCRIPTは、Adobe Systems社の登録商標です。
●UNIXオペレーティング・システムはAT&Tが開発し、ライセンスしています。
●SはAT&Tが開発し、ライセンスしています。

ソフトウェア技術社員募集中。
デモルーム開設中。

株式会社アイザック
〒150 東京都渋谷区渋谷1-11-8 渋谷清水第1ビル
TEL.03-406-1604(営業部直通) FAX.03-406-3910

資料請求 No.003H

KUBOTA COMPUTER INC.

オリジナルは、色褪せない。

カラーディスプレイが活きる。ネットワーキングが違う。

世界の主要なワークステーションに採用されている、MIPS社のR3000プロセッサ。RS3230は、MIPS社自らがそのR3000の能力を最大限に発揮するために開発したデスクトップ・ワークステーション。エンジニアリングだけでなく、パソコンとともにオフィスの幅広い業務にもご利用いただきたい汎用システムです。20MIPS、3.6MFLOPSのスペックをパソコン上位クラスの価格で実現するハイコストパフォーマンス。複数ウインドウでの漢字表示も余裕の高解像度カラーディスプレイ。大規模アプリケーションにも十分なメモリ容量最大128M、ディスク容量最大6.2GBをサポートします。もちろん、ネットワーキングは、ワークステーションの本領を発揮。主要なネットワークアーキテクチャ、通信プロトコルのほとんどをサポートするうえ、RS3230はすべてのMIPS社のワークステーション/サーバーとバイナリ互換を堅持しており、柔軟なコンピューティング環境が実現できます。

主な仕様●20MIPS（VAX MIPS）●SPECmack 17.3●3.6MFLOPS（LINPACK DP FORTRAN）●10万ベクトル/秒（X-Window上）●UNIX System Vおよび4.3BSD統合のRISC/os4.50、NFS4.0、BSD-TAHOEサポート●1280×1024、16/19インチカラーディスプレイ

＊RS3230, R3000は、MIPS Computer Systems Inc.の商標です。＊VAXは、Digital Equipment Corporationの商標です。＊X-Windowは、米国MITの商標です。＊NFSは、Sun Microsystems Inc.の商標です。＊UNIXは、米国AT&T社が開発し、ライセンスしているソフトウェアです。

標準構成

199万円（消費税別）

●カラー16インチディスプレイ ●200MBディスク ●8MBメモリ ●マウス ●キーボード ●OS等基本ソフトを含む

RS3230™

RS3230は日本ではクボタコンピュータ㈱が製造販売しています。

クボタコンピュータ株式会社 〒160 東京都新宿区新宿2-8-8 ☎03(5379)0661

大阪支店 〒541 大阪市中央区本町4-4-25 ☎06(264)2501　名古屋営業所 〒460 名古屋市中区丸の内2-18-14 KS-1ビル9階 ☎052(201)0561　山梨工場 〒400-02 山梨県中巨摩郡白根町下今諏訪907-8 ☎0552(84)4861

資料請求 No.004H

株式会社アスキー 〒107-24 東京都港区南青山6-11-1 スリーエフ南青山ビル ☎(03)486-7111(代表) ☎(03)498-0299(情報電話)

年ぎめ購読料8,160円（税・送料込） 振替 東京4-161144

定価680円（本体660円） Printed in Japan

雑誌05461-11

Super ASCII

コンピュータ技術総合誌

11

月刊スーパーアスキー

RISCはなぜ速いのか■PC-H98model 100/60■HyperCard 2.0

1990

ASCII